Linus Romer

The

FETAMONT

Typeface

Design and Constructions

October 30, 2015

Contents

1 Introduction

The logo font, known from logos like METAFONT or METAPOST, has been very limited in its collection of glyphs. The new typeface *Fetamont* extends the logo typeface in two ways:

- Fetamont consists of 256 glyphs, such that the T1 (a.k.a. EC, a.k.a. Cork) encoding table is complete now.
- Fetamont has additional faces like “light ultracondensed” or “script”.

The `fetamont` package provides LaTeX support for the Fetamont typeface. Both the package and the typeface are distributed on CTAN under the terms of the *LaTeX Project Public License* (LPPL).

This document describes the design and the constructions of the typeface itself. The LaTeX support for the Fetamont typeface is described in [Romer14].

2 Comparison With Existing Logos

The following picture shows the METAPOST and the METAFONT logos written in Fetamont (gray) and Taco Hoekwater’s Type 1 version of the logo font (outlined).

There are hardly any differences; only the "S" is significantly different, because its shape was changed by D. E. Knuth in 1997. The other faces of Hoekwater's *Logo* are also very similar to their corresponding Fetamont faces. Widths and kernings may rarely differ by one unit (except for the "A" in *Logo 9*, which has a strange width).

A comparison with the METATYPE1 logo from [Jackowski01] shows virtually no differences as well.[1]

The following picture compares *Fetamont Bold Condensed 40* with a traced version of the *Title Font* from `manfnt.mf`.

3 Naming Scheme For The Fetamont Faces

The file name of every face begins with the prefix `ffm`, which stands for «*f*ree typeface *f*eta*m*ont». The suffixes normally contain a symbol for the weight: `l` for light, `r` for regular, `b` for bold and `h` for heavy. The number at the end stands for the optical size (e.g. 10 pt). Depending on the face, the suffix is made of additional symbols:

Upright				Oblique			
	r8	b8	h8		o8	bo8	ho8
	r9	b9	h9		o9	bo9	ho9
l10	r10	b10	h10	lo10	o10	bo10	ho10
Condensed Upright				Condensed Oblique			
lc10	c10			lco10	co10		
		bc40				bco40	
Ultracondensed Upright				Ultracondensed Oblique			
lq10				lqo10			
Script Upright				Script Oblique			
lw10	w10	bw10	hw10	lwo10	wo10	bwo10	hwo10

[1] I have never seen the original sources of the "Y" and the "1" but I think that my imitated "Y" and "1" are extremely close to the original.

Section 7 shows the font tables of all these faces.

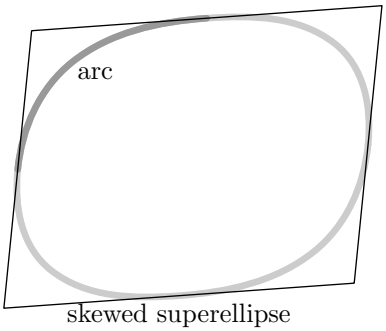
It is clear that thanks to the power of METAFONT the number of possible faces is theoretically endless. Anyone wishing to design a new face for Fetamont can do so by just redefining the parameters of `ffmr10.mf` and saving the file under a new name.

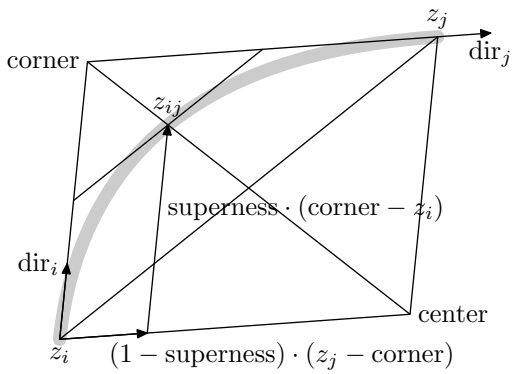
4 Special Techniques

Fetamont uses some special METAFONT techniques that are not well known (or have been unknown). The following subsections will document these techniques.

4.1 Arc Constructions

Practically all curved paths in *Fetamont* are made out of so-called *arcs*. An arc is a kind of a quarter of a skewed superellipse. The skew is only needed if the arcs have to look randomized like in the script style of fetamont.





In order to draw such an arc, the user defines the starting points z_i, the starting direction dir_i, the ending point z_j, the ending direction dir_j and a so-called *superness*. The macro **arc**$(z_i, \mathrm{dir}_i, z_j, \mathrm{dir}_j)$ then defines the path as follows:

- Compute the point z_{ij}, which is at center + superness · (corner − center) in vector terms. So if e.g. superness = 0.8, z_{ij} is reached after travelling 80 % of the straight path from corner to center. One can see easily, that z_{ij} can also be computed by

$$z_{ij} = z_i + \text{superness} \cdot (\text{corner} - z_i) + (1 - \text{superness}) \cdot (z_j - \text{corner})$$

- Now make a nice curve, that leaves z_i in the direction dir_i, passes z_{ij} in the direction $z_j - z_i$ and ends in z_j heading for the direction dir_j.

Here is the METAFONT translation of this construction report:

```
vardef arc(expr zi,diri,zj,dirj) =
 zi{diri}...
 begingroup
```

```
  save corner,zij;
  pair corner,zij;
  corner=zi+whatever*diri=zj+whatever*dirj;
  zij=zi
   +superness*(corner-zi)
   +(1-superness)*(zj-corner);
  zij
 endgroup{zj-zi}
 ...zj{dirj}
enddef;
```

Everything in between `begingroup` and `endgroup` is just the computation of z_{ij}.

Note that Donald E. Knuth used a little different approach to draw randomized arcs for his «crazy shapes» of the Logo typeface.

4.2 Glyph names

Plain METAFONT automatically assigns well known letters like `"A"` with the corresponding encoding slot 65. But this does not work for letters like `"Ä"` (nor `"Adieresis"`) as these letters will be placed in different encoding slots depending on the encoding. So these letters have to be declared directly by its encoding number (code). However, this will become problematic if one wants to change the encoding.

I solved this problem by a macro `enc` that uses very long conditionals to assign a unique code to each unicode name:

```
def enc(expr name)=
 if (font_coding_scheme_="T1"):
  if name="grave":
   0
  elseif name="acute":
   1
  elseif name="circumflex":
   2
  ...
  elseif name="germandbls":
   255
  else:
   errmessage("unknown name to encode");
  fi
 else:
  errmessage("tell me somewhere that the font_coding_scheme is T1");
 fi
enddef;
```

One may think that this is a very bad programming style and that a macro using arrays

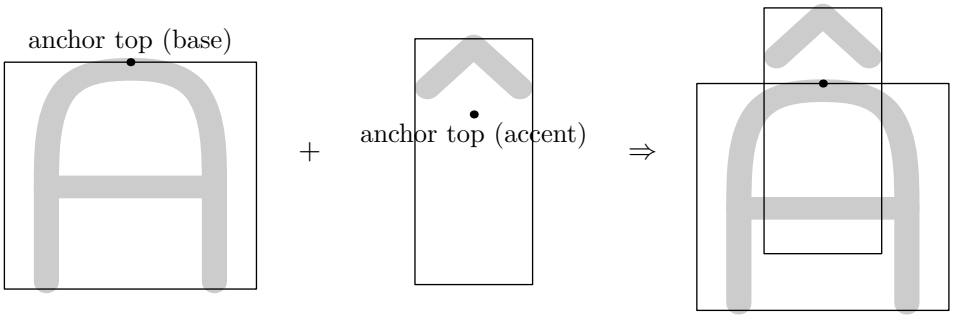
would be much more elegant. I agree! But then again I have found this to be the fastest solution.

With the `enc` macro one can treat «A» and «Ä» equally: `enc("A")` = 65 and `enc("Adieresis")` = 196.

4.3 Combined Characters

In order to draw accented and other combined characters, it is helpful to use *anchors*. The concept of anchors is common in type design outside of the METAFONT world. However, anchors rarely have been seen in METAFONT up to now.

The idea is easy: Put an anchor at a given point in a base glyph and in the accent glyph; then overlay the two glyphs such that the anchors coincide, producing the pre-composed accented character.



Normally there are several kind of anchors needed. E.g. «Â» and «Ą» need two different anchors and so do «Ĺ» and «Ł». Fetamont needs three kind of anchors: «top», «topright» and «bot». So there are three arrays that can store the anchors:[2]

```
pair charanchortops_[];
pair charanchorbots_[];
pair charanchortoprights_[];
```

If one writes `charanchortops_[charcode]=(.5w,h);` one stores a «top» anchor for the current glyph at the point $(.5w, h)$. Of course one needs more information, so there exist additional arrays:

```
numeric charwidths_[];
numeric charheights_[];
numeric chardepths_[];
numeric charitalcorrs_[];
picture charpictures_[];
```

The empty places in these arrays are always automatically filled in at the end of each character:

[2]There is a naming convention that symbolic tokens ending in «_» should not be used in high level programming.

```
extra_endchar:=extra_endchar&"charpictures_[charcode]:=currentpicture;"
 &"charwidths_[charcode]=charwd;"
 &"charheights_[charcode]=charht;"
 &"chardepths_[charcode]=chardp;"
 &"charitalcorrs_[charcode]=charic;";
```

You can now combine two characters with the macro `ffmcombinedchar` which takes the following parameters:

namea	name of the base character
nameb	name of the accent character
namec	name of the new (combined) character
anchor	name of the anchor
height	new total height
depth	new total depth

The definition of the macro is now quite straight forward. The `code_offset` is needed, because the same constructions are used twice, as the lowercase letters are formed by small capitals.

```
def ffmcombinedchar(expr namea,nameb,namec,anchor,height,depth) =
 beginchar(enc(namea)+code_offset,
  charwidths_[enc(nameb)+code_offset],height,depth);
 charic:=charitalcorrs_[enc(nameb)+code_offset];
 addto currentpicture also charpictures_[enc(nameb)+code_offset];
 if anchor="top":
  addto currentpicture also charpictures_[enc(namec)] shifted
  ((charanchortops_[enc(nameb)+code_offset]
   -charanchortops_[enc(namec)]) slanted slant);
 elseif anchor="bot":
  addto currentpicture also charpictures_[enc(namec)] shifted
  ((charanchorbots_[enc(nameb)+code_offset]
   -charanchorbots_[enc(namec)]) slanted slant);
 elseif anchor="topright":
  addto currentpicture also charpictures_[enc(namec)] shifted
  ((charanchortoprights_[enc(nameb)+code_offset]
   -charanchortoprights_[enc(namec)]) slanted slant);
 else:
  errmessage "Wrong anchor name";
 fi
 endchar;
enddef;
```

4.4 Kerning Classes with METAFONT

Like anchor positioning, the concept of kerning classes is widely known but not frequently used in METAFONT. The reason for this is that METAFONT cannot natively write

kernings for multiple characters at once. Hence, multiple kerning information has to be cached in arrays.

It is clear that "OT" needs the same kerning as "DT". But be aware, "TO" needs a different kerning as "TD"! So there are two kind of kerning classes:

- *first kerning classes* group glyphs together that share the same shape to the right like "D" and "O"
- *second kerning classes* group glyphs together that share the same shape to the left like "C" and "O"

We define the arrays `kernclassesf_[][]` and `kernclassess_[][]` to store this information:

```
numeric kernclassesf_[][],
 kernclassess_[][],
 ligmatrix_[][][];
```

The third array called `ligmatrix` will store all relevant kerning and ligature information. Now

```
 addkernclassf("V","W");
 addkernclasss("T","Tcaron","Tcedilla");
```

will group "V" and "W" to a first kerning class and "T", "Tcaron" and "Tcedilla" to a second kerning class. The definitions of the macros `addkernclassf` and `addkernclasss` are analogous, they just deal with different arrays.

METAFONT has no straight way to determine the length of arrays or subarrays, so these lengths have to be stored somewhere. Thus, the zeroth row of the array consists of only one item: `kernclassesf_[0][0]` stores the number of rows (which corresponds to the number of first kerningclasses). Each kerning class is stored in a row. The zeroth item of these rows is always the length of the row (which corresponds the number of glyphs in the kerning class).

```
def addkernclassf(text a) =
 kernclassesf_[0][0]:=kernclassesf_[0][0]+1; % number of kernclassesf
 begingroup
  save i;
  i:=0; % number of chars in current class
  for b=a:
   i:=i+1;
   kernclassesf_[kernclassesf_[0][0]][i]:=enc(b);
  endfor
  % number of chars in current class is stored at 0th position
  kernclassesf_[kernclassesf_[0][0]][0]:=i;
 endgroup
enddef;
```

The macros `addclasskern` and `addlig` will now add kerning information to kerning classes or add ligatures for single glyphs, respectively. This information is stored in the `ligmatrix_[][][]`. In order to understand the definitions of the macros `addclasskern` and `addlig` it is important to know how this storage works:

For every glyph number of the encoding (from 0 to 255) the array `ligmatrix_[][][]` has a subarray reserved, so `ligmatrix_[][][]` consists of 256 rows. Each row contains the complete kerning and ligature data for the glyph whose encoding number equals the row number.

Let us say that the glyph number f = 102 shall be kerned together with a = 97 by the amount of $-.5u\#$ and kerned together with t = 116 by the amount of $u\#$. Furthermore f shall be combined with l = 108 to the ligature fl = 29. So the 102th row will hold this information as follows:

$$\text{ligmatrix}_{102} = (\ \underbrace{3}_{\text{length}}\ , \underbrace{(97, -.5u\#)}_{\text{kern with a}}, \underbrace{(116, u\#)}_{\text{kern with t}}\ , \underbrace{(-108, 29)}_{\text{ligature fl}}\ )$$

The minus flag before glyph numbers distincts ligatures from kernings.

At the beginning, the ligmatrix is empty, so each row has length 0 which is stored at the zeroth position of the rows:

```
for i=0 upto 255:
 ligmatrix_[i][0][0]:=0;
endfor
```

The call `addclasskern("f","a",-.5u#)` will kern the first kerning class that contains «f» as first item and the second kerning class that contains «a» as first item by the amount of $-.5u\#$. The macro `addclasskern` writes the kerning information directly into the `ligmatrix_[][][]` for all class members, the only problem is to find the indices of the kerning classes:

```
def addclasskern(expr first,second,kvalue) =
 begingroup
  save i,j,m,n;
  % get the indices i and j of the two classes:
  i:=0; % default value (cannot be true)
  j:=0; % default value (cannot be true)
  forever:
   i:=i+1;
   exitif kernclassesf_[i][1]=enc(first);
   if i>255:
    errmessage("unknown first kerning class");
   fi
  endfor
  forever:
   j:=j+1;
```

```
   exitif kernclassess_[j][1]=enc(second);
   if j>255:
    errmessage("unknown first kerning class");
   fi
  endfor
  for k=1 upto kernclassesf_[i][0]:
   m:=kernclassesf_[i][k]; % current first glyph
   for l=1 upto kernclassess_[j][0]:
    ligmatrix_[m][0][0]:=ligmatrix_[m][0][0]+1;
    n:=ligmatrix_[m][0][0]; % current last entry index (being written)
    ligmatrix_[m][n][0]:=kernclassess_[j][l];
    ligmatrix_[m][n][1]:=kvalue;
   endfor
  endfor
 endgroup
enddef;
```

The call `addlig("f","l","fl")` stores in the `ligmatrix_[][][]` the instruction, that the combination of «f» and «l» shall be replaced by the «fl» ligature:

```
def addlig(expr first,second,third) =
 begingroup
  save i,n;
  i:=enc(first); % encoding number of first
  ligmatrix_[i][0][0]:=ligmatrix_[i][0][0]+1;
  n:=ligmatrix_[i][0][0]; % current last ligature entry index of i
  ligmatrix_[i][0][0]:=n;
  ligmatrix_[i][n][0]:=-enc(second); % minus is a flag for "ligature"
  ligmatrix_[i][n][1]:=enc(third);
 endgroup
enddef;
```

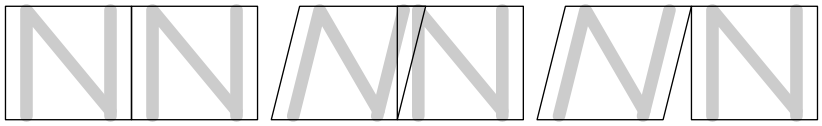
At the very end, the macro `writeligtable` writes all information from `ligmatrix_[][][]` in a METAFONT friendly way:

```
def writeligtable = % write all kernings/ligatures at once
 begingroup
  save n;
  for i=0 upto 255: % current glyph i
   n:=ligmatrix_[i][0][0]; % number n of entries  for glyph i
   if n<>0: % skip empty entries
    ligtable i:
    for j=1 upto n-1: %last entry needs a semicolon
     if ligmatrix_[i][j][0]<0: % the minus is a flag for "ligature"
      -ligmatrix_[i][j][0]=:ligmatrix_[i][j][1],
```

```
     else:
      ligmatrix_[i][j][0] kern ligmatrix_[i][j][1],
     fi
    endfor
    %last entry needs a semicolon:
    if ligmatrix_[i][n][0]<0: % the minus is a flag for "ligature"
     -ligmatrix_[i][n][0]=:ligmatrix_[i][n][1];
    else:
     ligmatrix_[i][n][0] kern ligmatrix_[i][n][1];
    fi
   fi
  endfor
 endgroup
enddef;
```

4.5 Italic Corrections

Letter spacing is unproblematic if two upright letters are combined, like «NN». But if the first letter is italic, the letters will get too close (like «*N*N») and need additional space (like «*N*N»). This additional space is called *italic correction*.



D. E. Knuth has already defined an italic correction for the letter «T», because this is the last letter of the logos METAFONT and METAPOST. As for the *Computer Modern* typeface he found `italcorr ht#*slant+.5u#` to be a suitable italic correction. However, this is not a perfect idea because the italic correction should tend to 0 (and not `.5u#`) when the slant tends to 0. Hence, every character in Fetamont different to «T» has an italic correction proportional to the slant and the letter height. E.g. the letter «A» has an italic correction of `.8ht#*slant`.

4.6 Producing Outlines

The METAFONT sources have been converted to outline font formats like Type 1 or OpenType by a python script. This script calls METAPOST to produce PostScript files for each glyph. These glyphs are imported by the `fontforge` module. Khaled Hosny already used this technique in [Hosny11] to produce the outlines of *Punk Nova*. Because the glyph widths get lost by importing, also the `tfm` module from the mftrace project is needed (see [Nienhuys06]).

The following script contains the most important parts of the conversion.

```
#!/usr/bin/python
```

```
import os
import sys
import fontforge
import tfm # this is tfm.py from mftrace
import glob
import subprocess
import tempfile
import shutil

def usage():
    print "Example␣usage:␣%s␣mysource" % sys.argv[0]

if __name__ == "__main__":
        if len(sys.argv) < 2:
                usage()
                sys.exit()

        print "Creating␣font␣file..."
        style = sys.argv[1]
        designsize = 10
        fontname = sys.argv[1]
        font      = fontforge.font()

        print "Setting␣general␣font␣information..."
        fontforge.loadEncodingFile("t1.enc")
        font.encoding="T1Encoding"

        print "Running␣METAPOST␣for␣tfm␣and␣glyphs␣definition..."
        mffile = os.path.abspath("%s" % fontname)
        tempdir = tempfile.mkdtemp()
        magnification = 1003.75/designsize
        subprocess.call(
                ['mpost',
                '&mfplain',
                '\mode=localfont;',
                'mag:=%s;' % magnification,
                'outputtemplate:="%c.eps";',
                'input␣%s;' % mffile,
                'bye'],
                stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                cwd=tempdir,
                )

        print "Importing␣glyphs..."
```

```
glyph_files = glob.glob(os.path.join(tempdir, "*.eps"))
for file in glyph_files:
        code  = int(os.path.splitext(os.path.basename(file))[0])
        glyph = font.createMappedChar(code)
        glyph.importOutlines(file, ("toobigwarn", "correctdir"))

print "Adding␣metrics..."
metric = tfm.read_tfm_file ("%s/%s.tfm" % (tempdir, fontname) )
for glyph in font.glyphs():
        metric_width = metric.get_char(glyph.encoding).width
        glyph.width = int (round (metric_width / designsize * 1000))
font.mergeFeature("%s/%s.tfm" % (tempdir, fontname))
shutil.copyfile("%s/%s.tfm" % (tempdir, fontname), "%s.tfm" % fontname)

shutil.rmtree(tempdir)

print "Add␣space␣for␣non-TeX..."
normal_space = font[32].width ##take width from visible space
font.encoding = "unicode"
font.createChar(32)
font[32].width = normal_space ##space
font.encoding = "T1Encoding"
font.encoding = "compacted"

print "Finetuning..."
font.selection.all()
font.addExtrema()
font.removeOverlap()
font.simplify()
font.round()
font.simplify()
font.autoHint()

print "Saving␣sfd-file␣'%s'..." % fontname
font.save("%s.sfd" % fontname)
print "Generating␣otf-file␣'%s'..." % fontname
font.generate("%s.otf" %fontname)
```

4.7 Randomize Feature

Normally, the randomization of the script faces has a fixed seed. However, for the OpenType versions of the script faces I have additionaly included five variants with random seeds. ConTEXt/LuaTEX can access these variants via the Randomize feature.

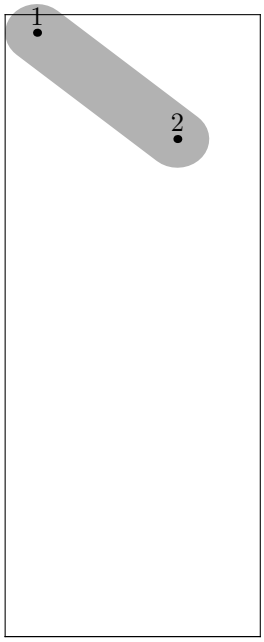
Eben schließt in sanfter Ruh
Lämpel seine Kirche zu;
Und mit Buch und Notenheften
Nach besorgten Amtsgeschäften,
Lenkt er freudig seine Schritte
Zu der heimatlichen Hütte,
Zündet er sein Pfeifchen an.

The text shown above is the product of the following source:

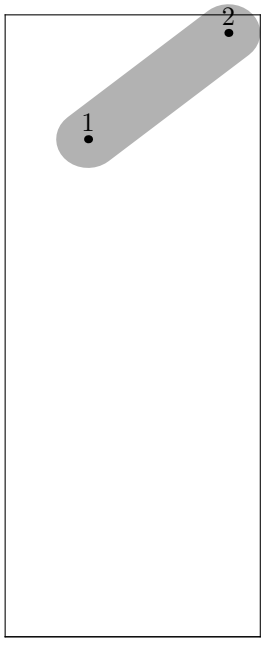
```
\definefontfeature[ffm][mode=node,script=latn,kern=yes,
liga=yes,rand=yes,dlig=yes]
\starttypescript [serif] [fetamont]
\definefontsynonym[Serif][file:ffmw10][features=ffm]
\definefontsynonym[SerifBold][file:ffmbw10][features=ffm]
\stoptypescript
\starttypescript [fetamont]
\definetypeface [fetamont] [rm] [serif] [fetamont] [default]
\stoptypescript
\setupbodyfont[fetamont]
\starttext
\rm
{\bf Eben schließt in sanfter Ruh}\\
Lämpel seine Kirche zu;\\
Und mit Buch und Notenheften\\
Nach besorgten Amtsgeschäften,\\
Lenkt er freudig seine Schritte\\
Zu der heimatlichen Hütte,\\
Zündet er sein Pfeifchen an.
\stoptext
```

5 Proof Sheets Of All Glyphs

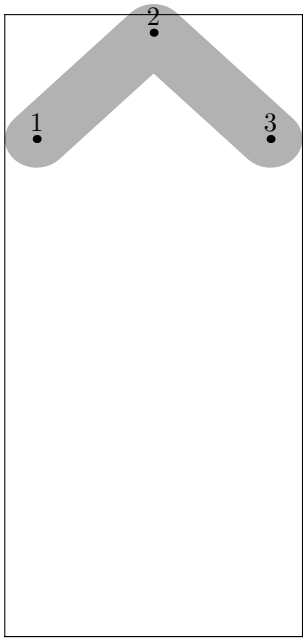
Glyph with coding number 0



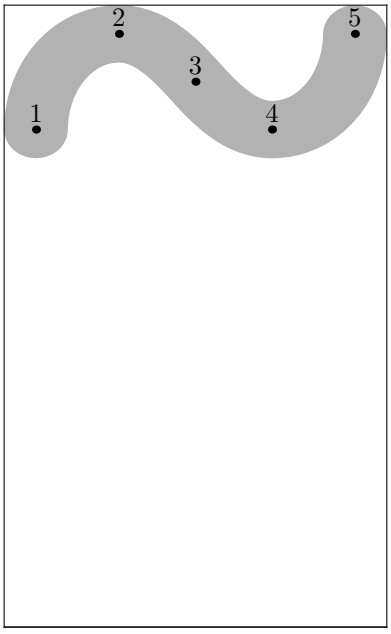
Glyph with coding number 1



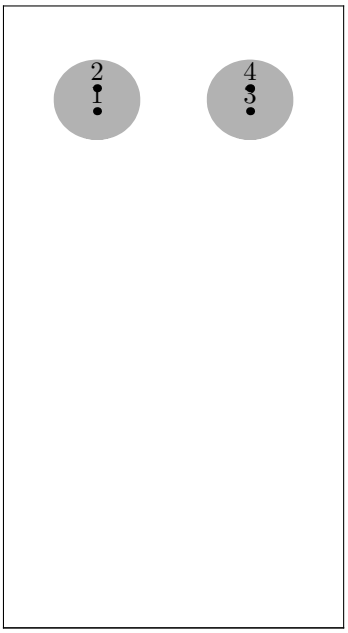
Glyph with coding number 2



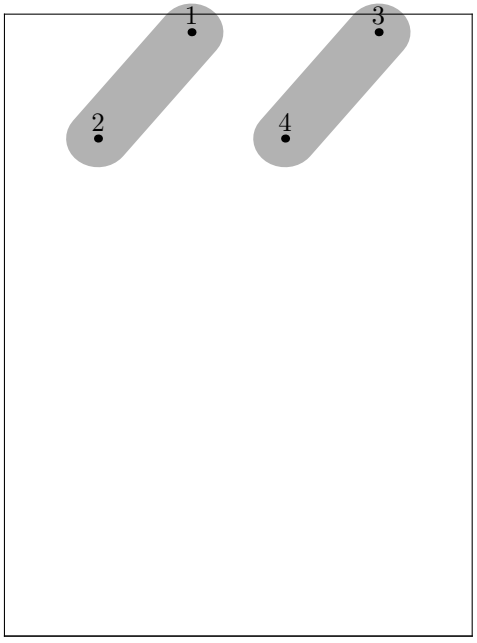
Glyph with coding number 3



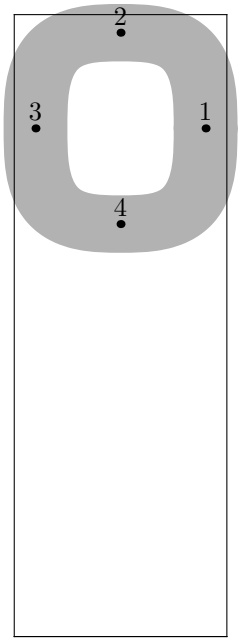
Glyph with coding number 4



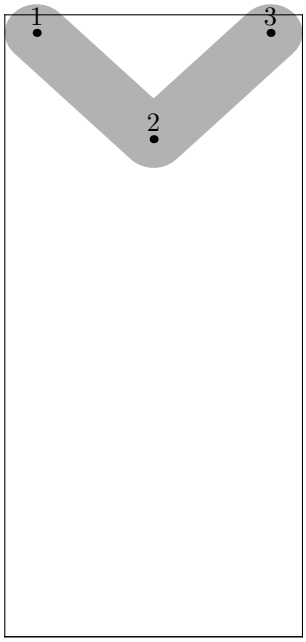
Glyph with coding number 5



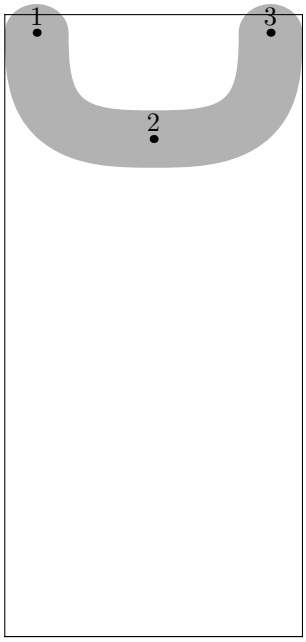
Glyph with coding number 6



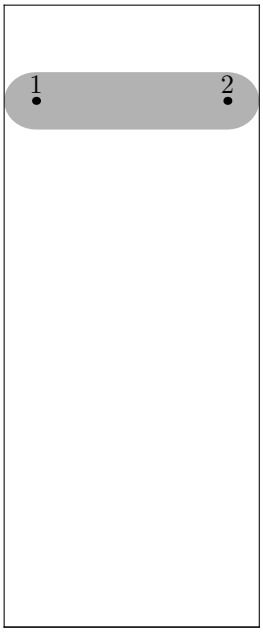
Glyph with coding number 7



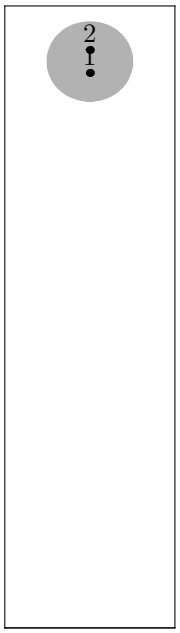
Glyph with coding number 8



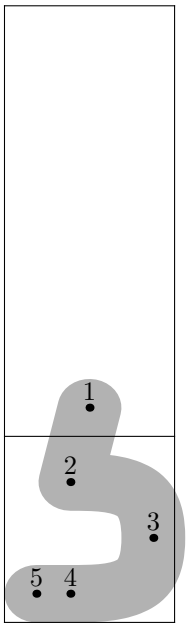
Glyph with coding number 9



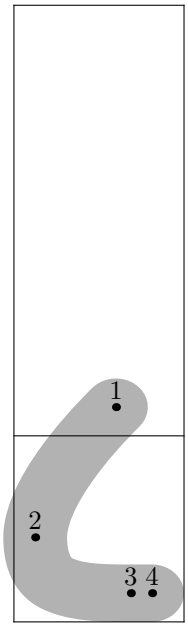
Glyph with coding number 10



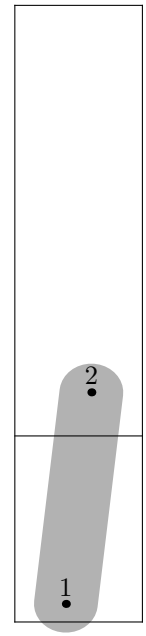
Glyph with coding number 11



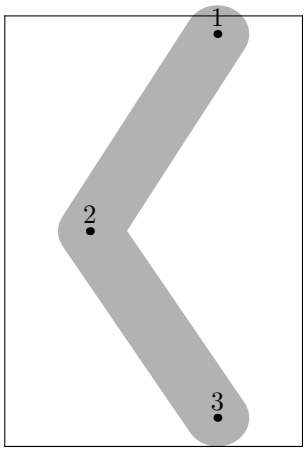
Glyph with coding number 12



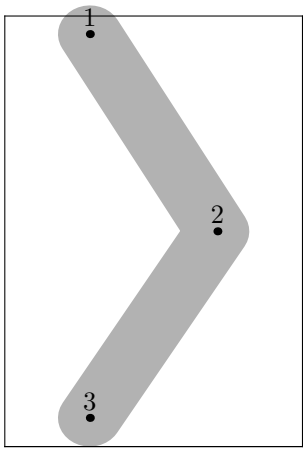
Glyph with coding number 13



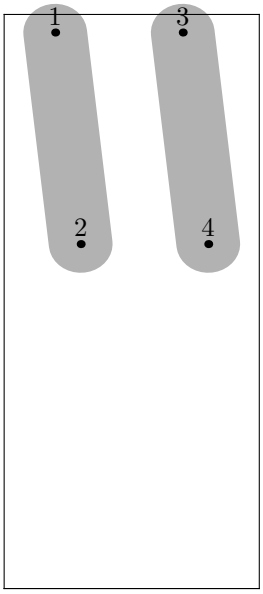
Glyph with coding number 14



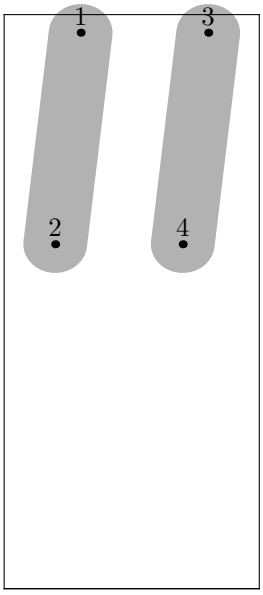
Glyph with coding number 15



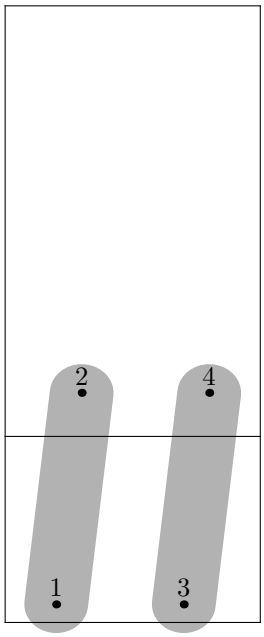
Glyph with coding number 16



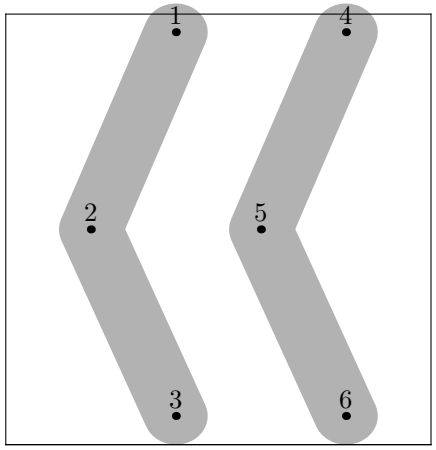
Glyph with coding number 17



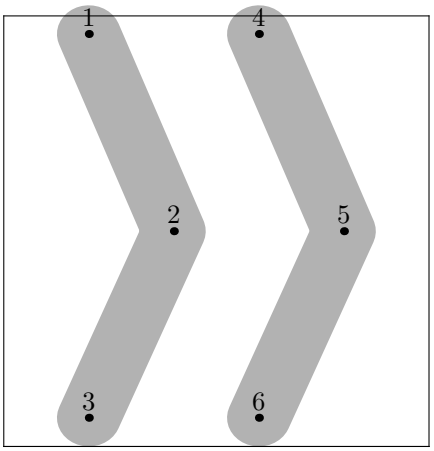
Glyph with coding number 18



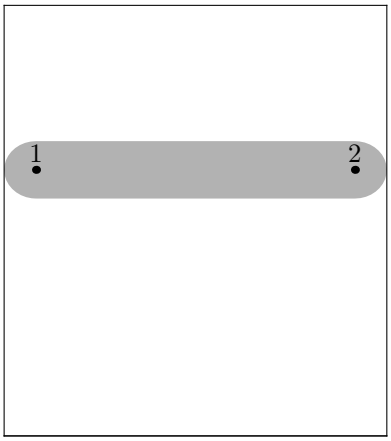
Glyph with coding number 19



Glyph with coding number 20



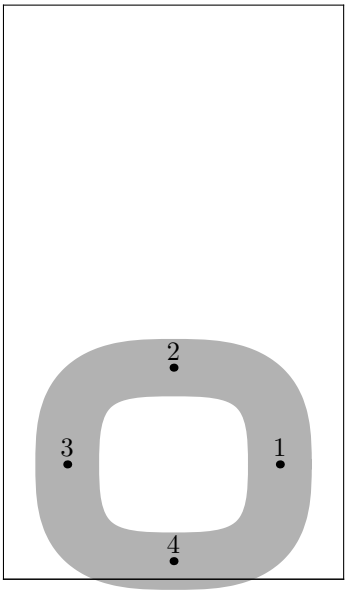
Glyph with coding number 21



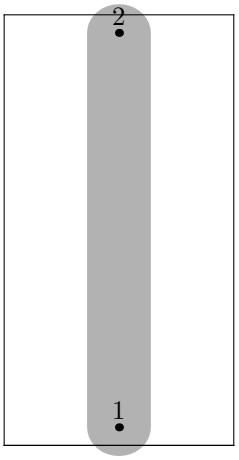
Glyph with coding number 22

Glyph with coding number 23

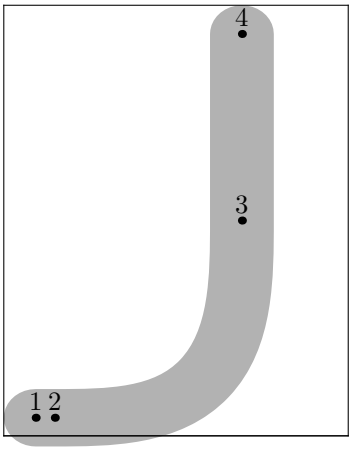
Glyph with coding number 24



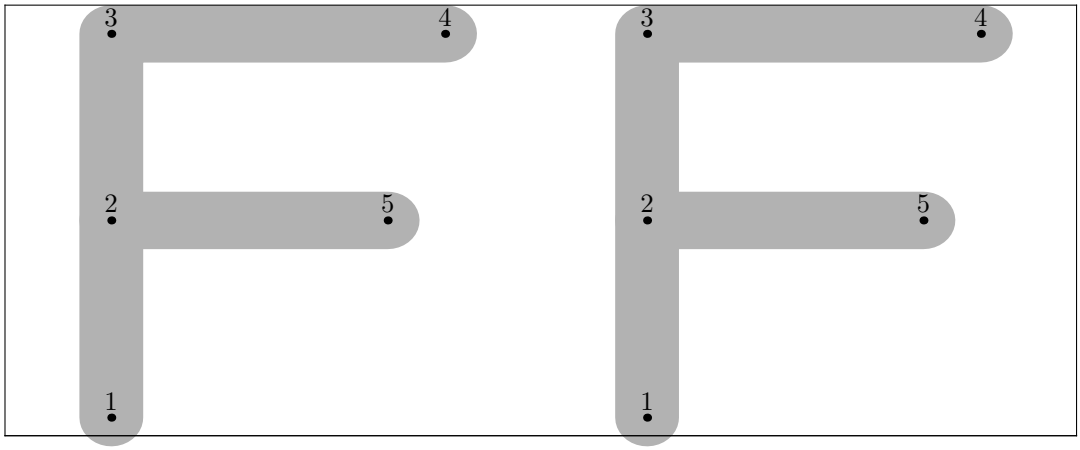
Glyph with coding number 25



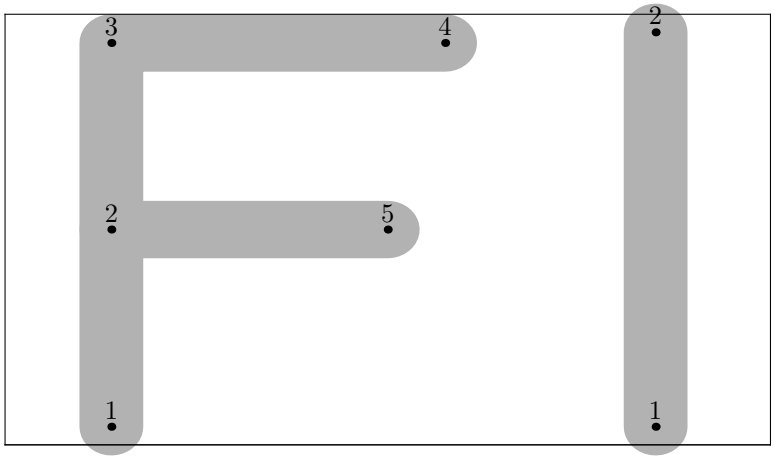
Glyph with coding number 26



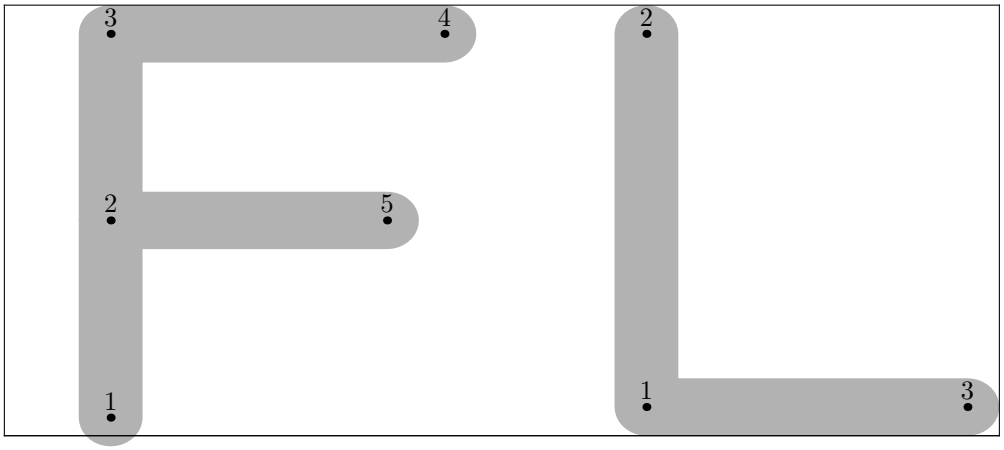
Glyph with coding number 27



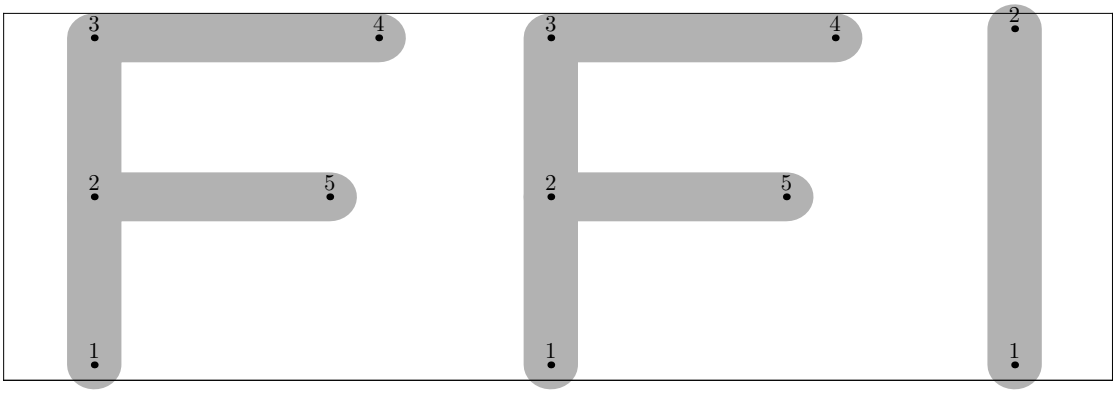
Glyph with coding number 28



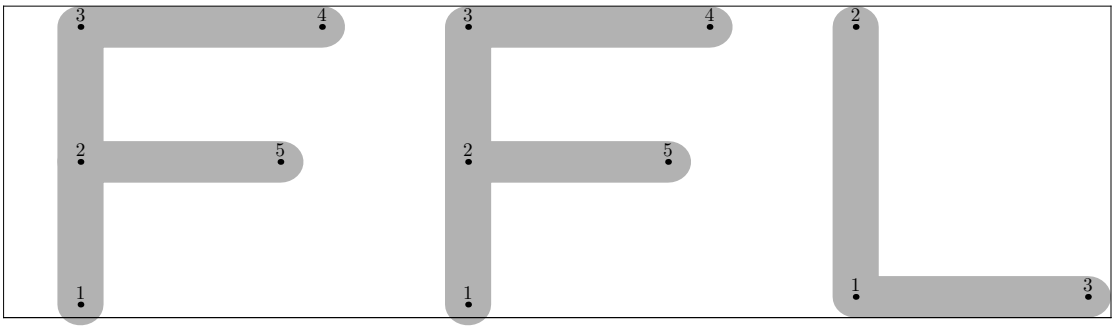
Glyph with coding number 29



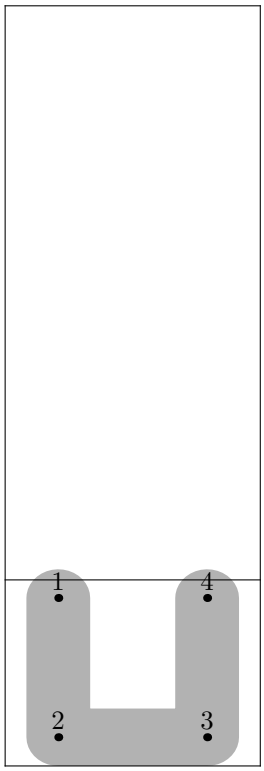
Glyph with coding number 30



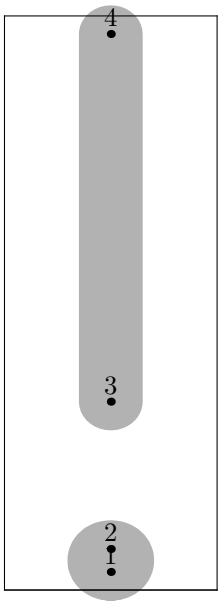
Glyph with coding number 31



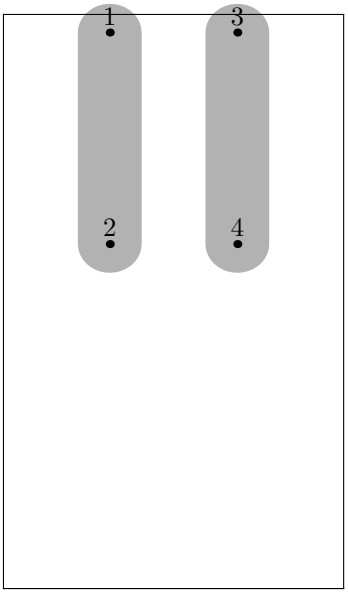
Glyph with coding number 32



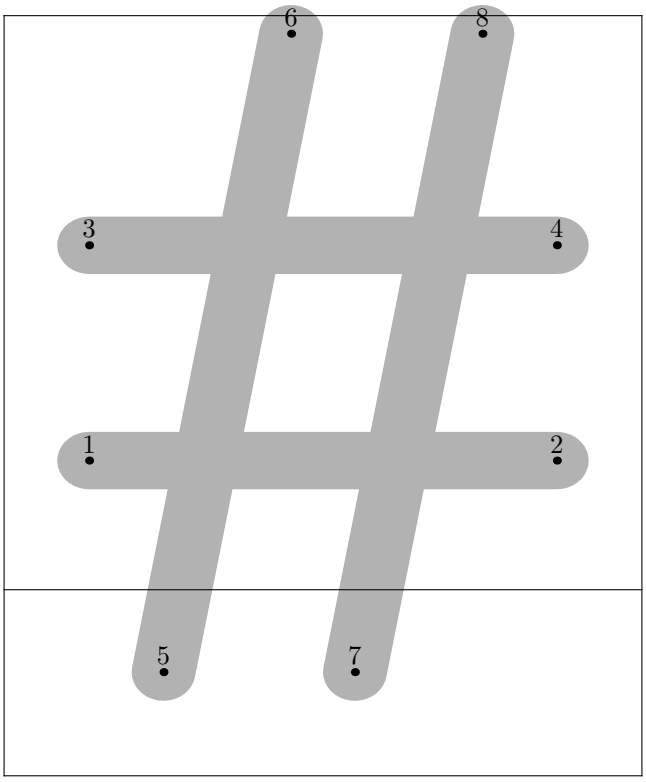
Glyph with coding number 33



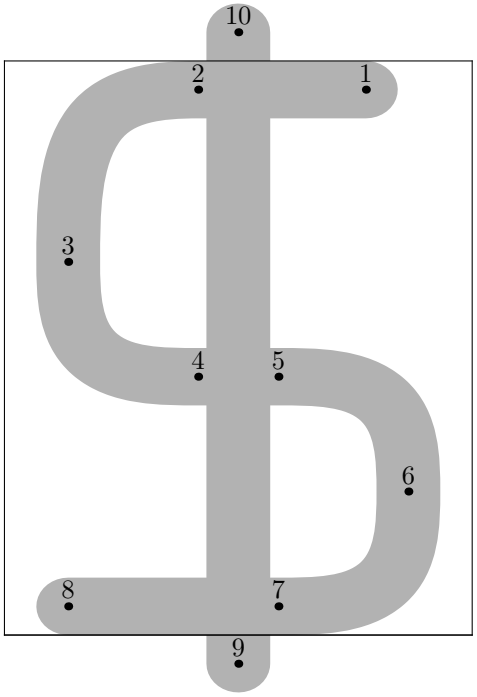
Glyph with coding number 34



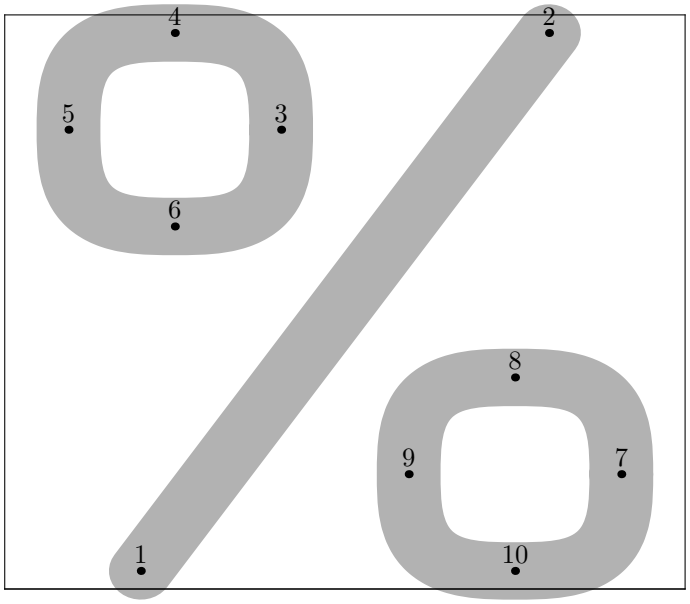
Glyph with coding number 35



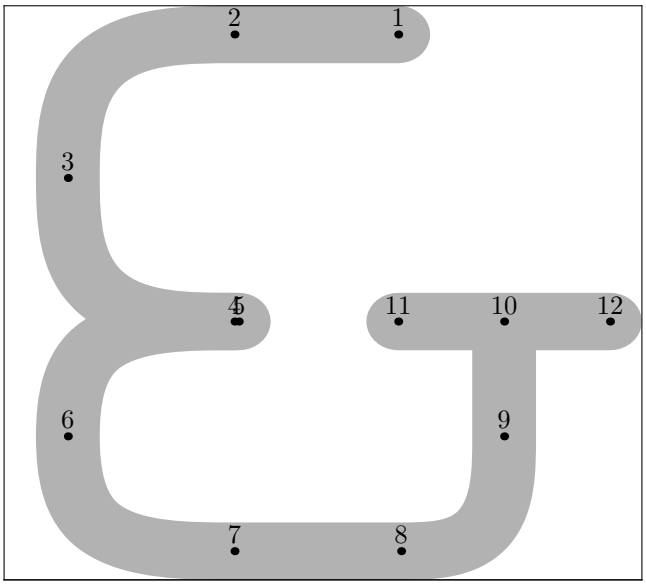
Glyph with coding number 36



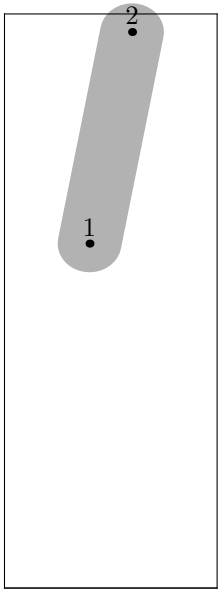
Glyph with coding number 37



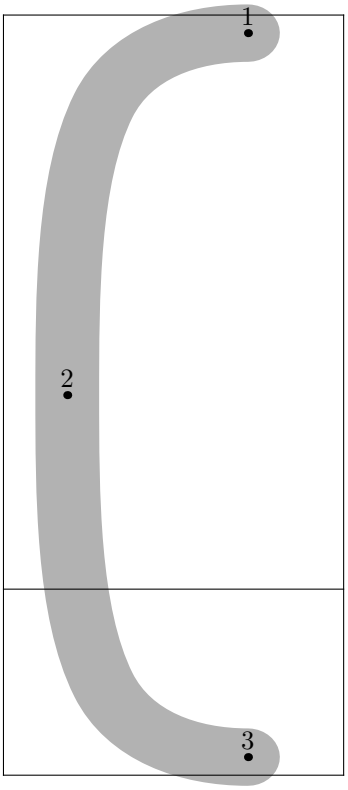
Glyph with coding number 38



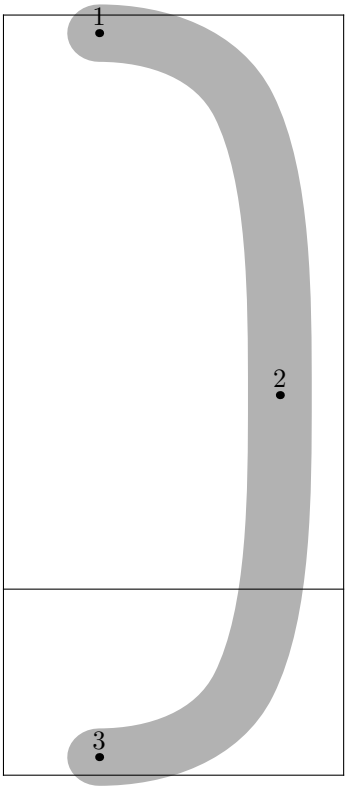
Glyph with coding number 39



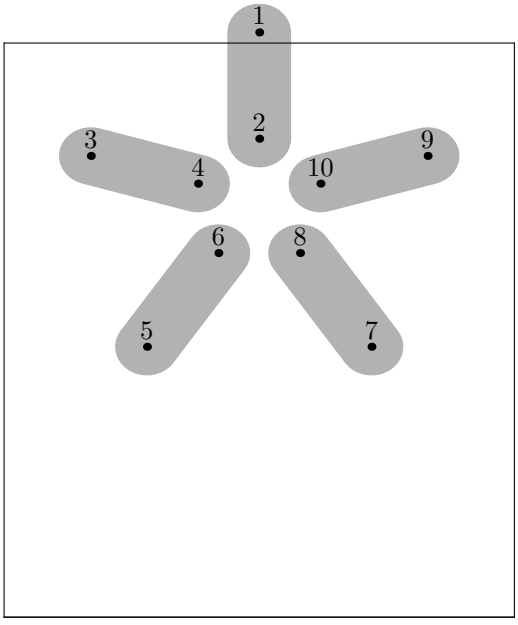
Glyph with coding number 40



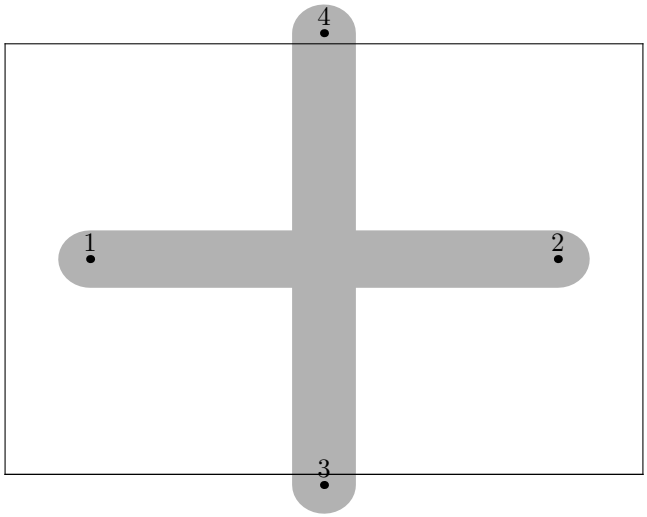
Glyph with coding number 41



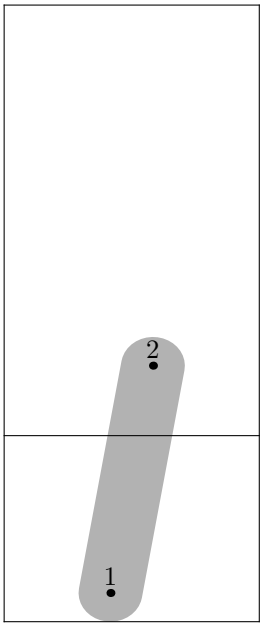
Glyph with coding number 42



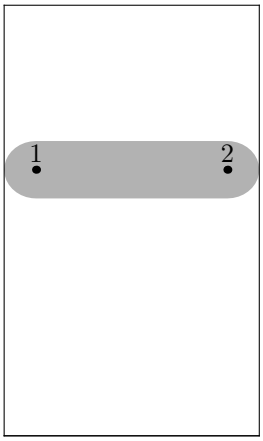
Glyph with coding number 43



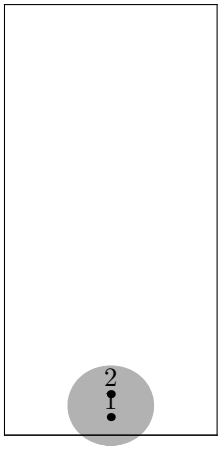
Glyph with coding number 44



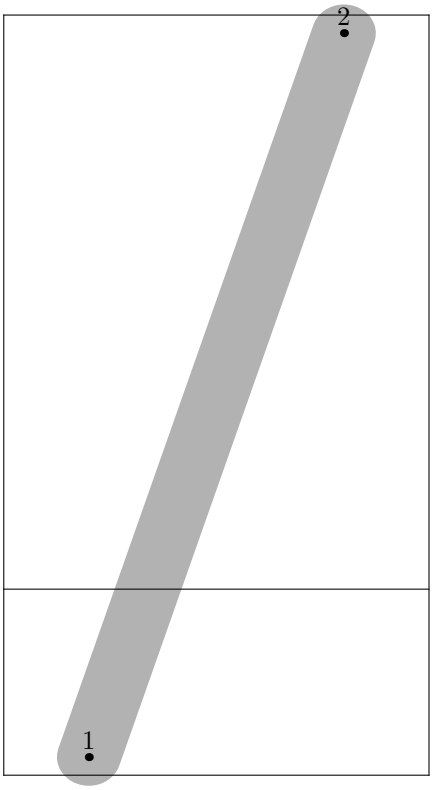
Glyph with coding number 45



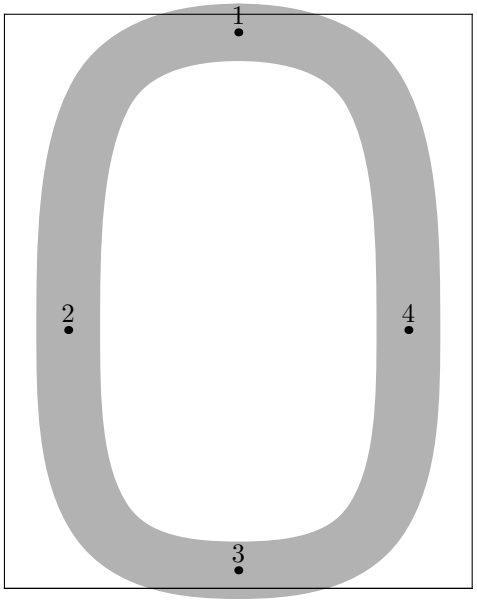
Glyph with coding number 46



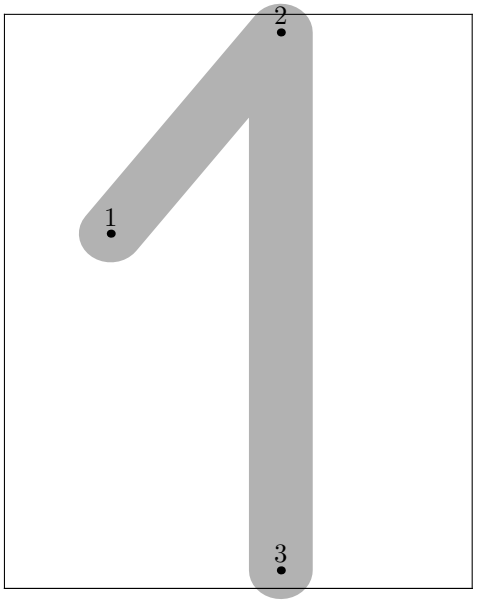
Glyph with coding number 47



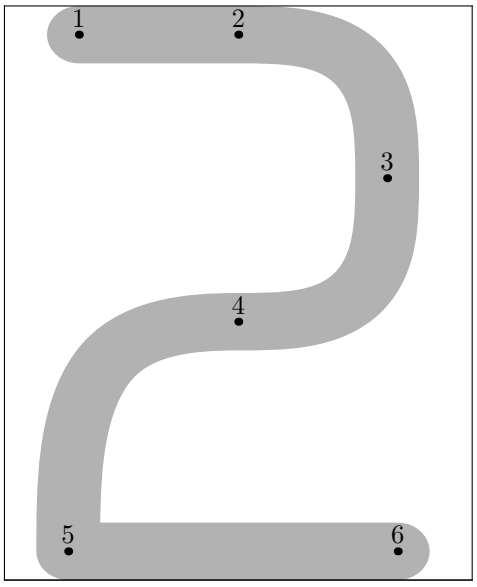
Glyph with coding number 48



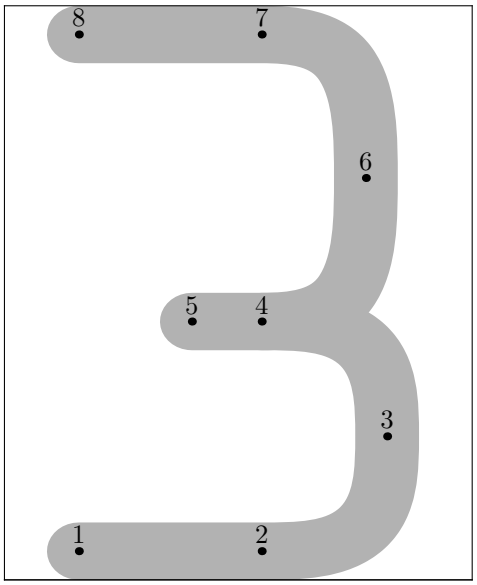
Glyph with coding number 49



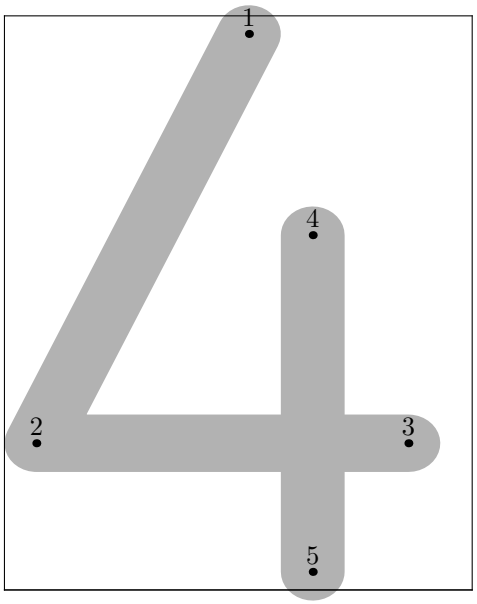
Glyph with coding number 50



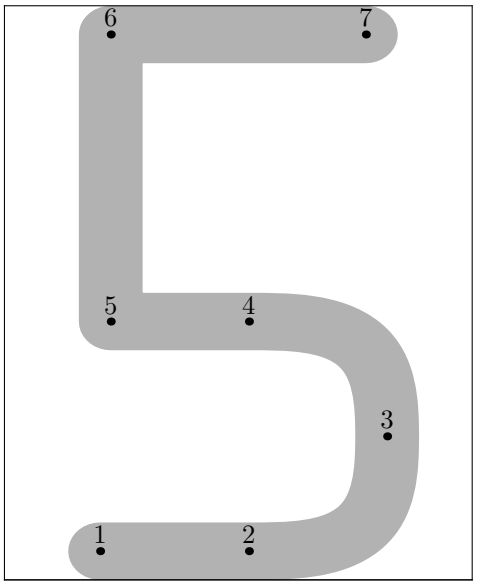
Glyph with coding number 51



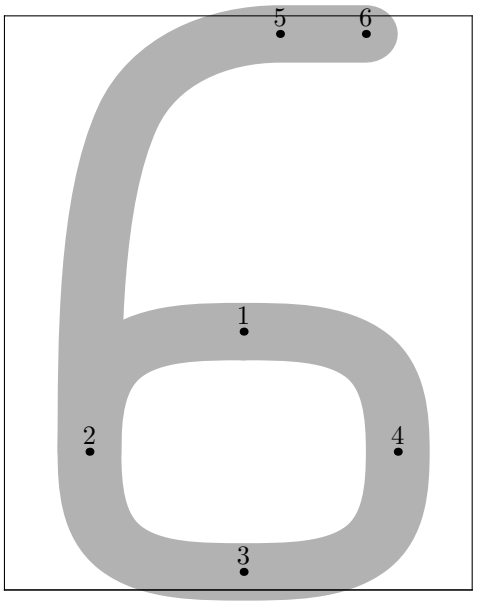
Glyph with coding number 52



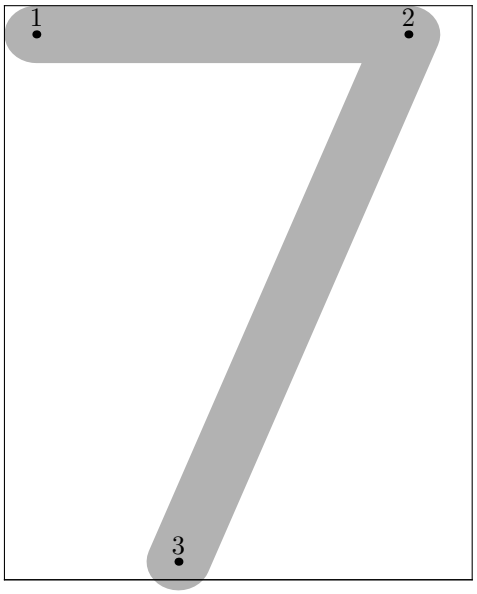
Glyph with coding number 53



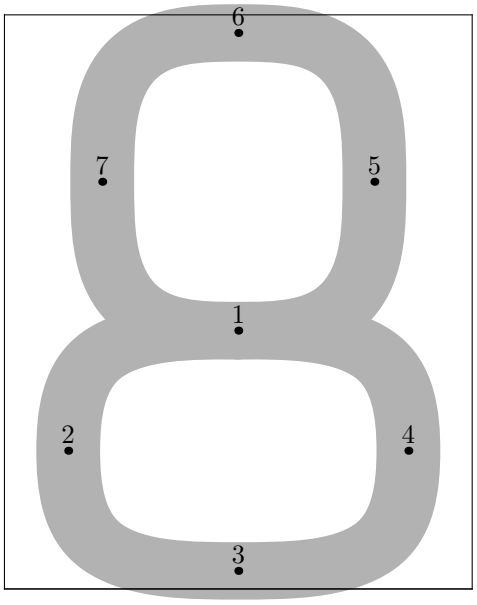
Glyph with coding number 54



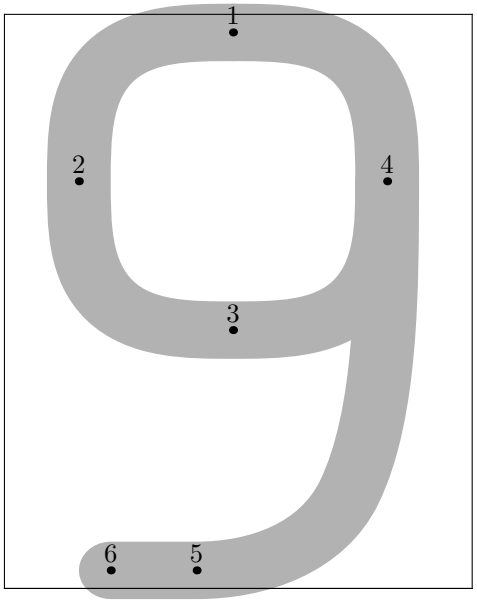
Glyph with coding number 55



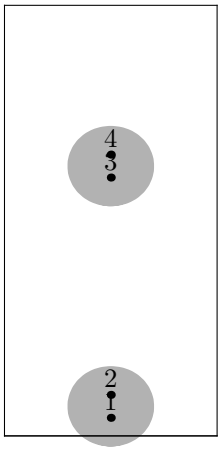
Glyph with coding number 56



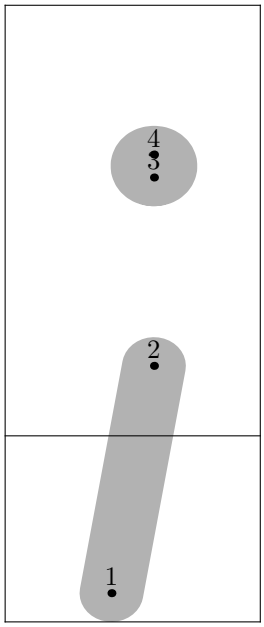
Glyph with coding number 57



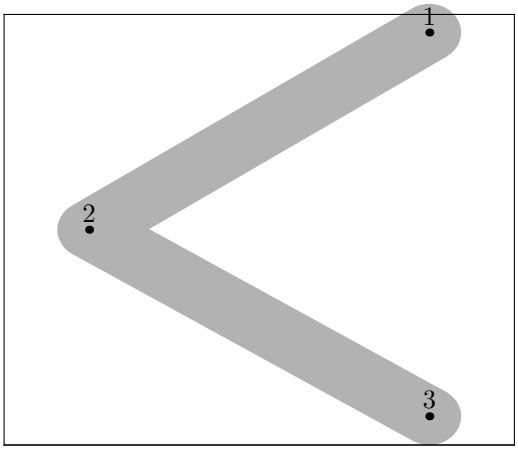
Glyph with coding number 58



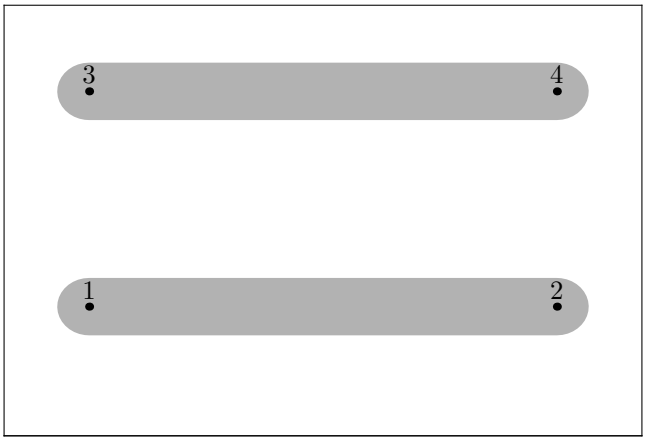
Glyph with coding number 59



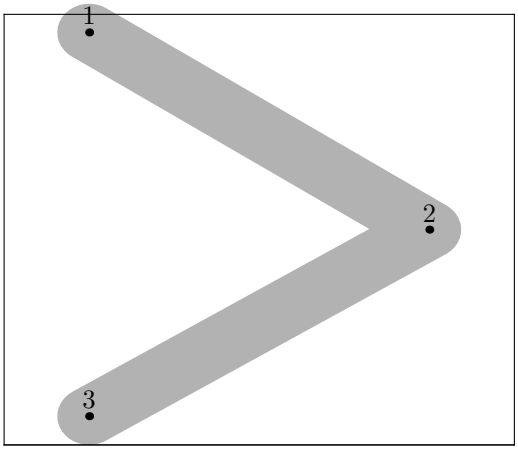
Glyph with coding number 60



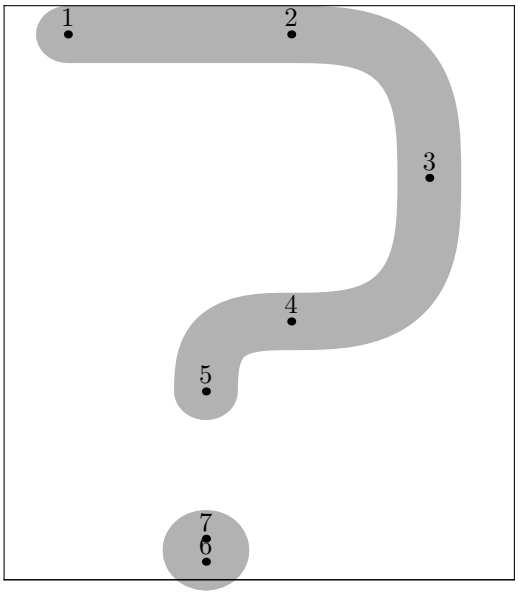
Glyph with coding number 61



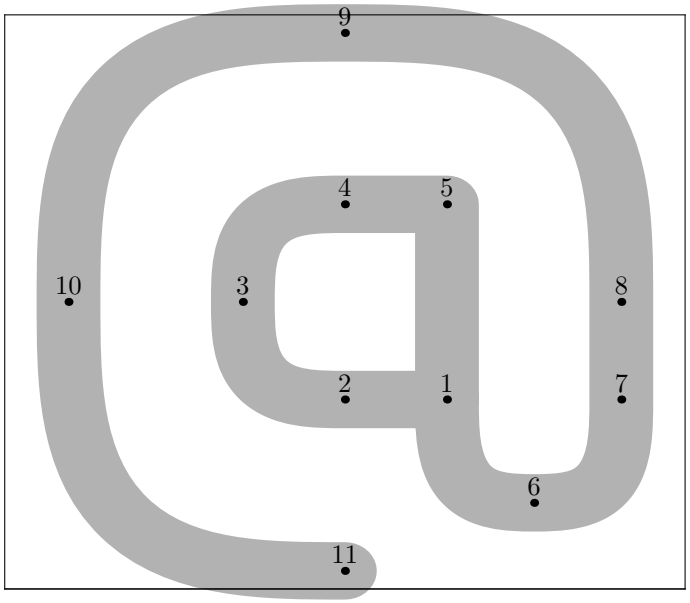
Glyph with coding number 62



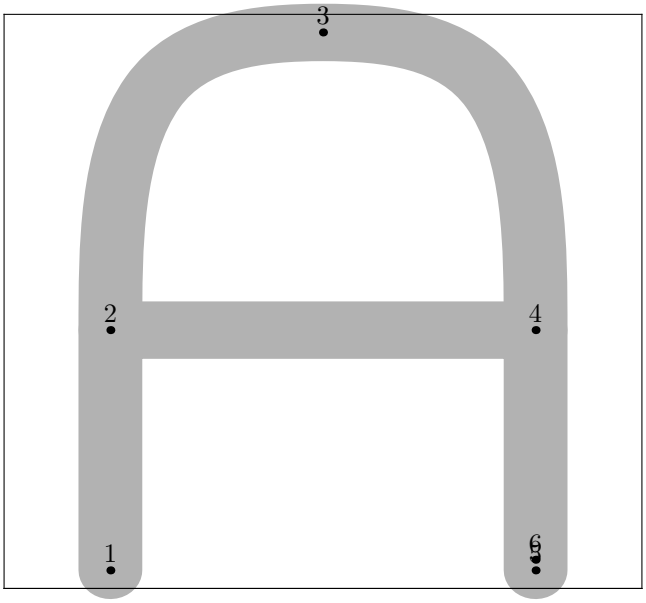
Glyph with coding number 63



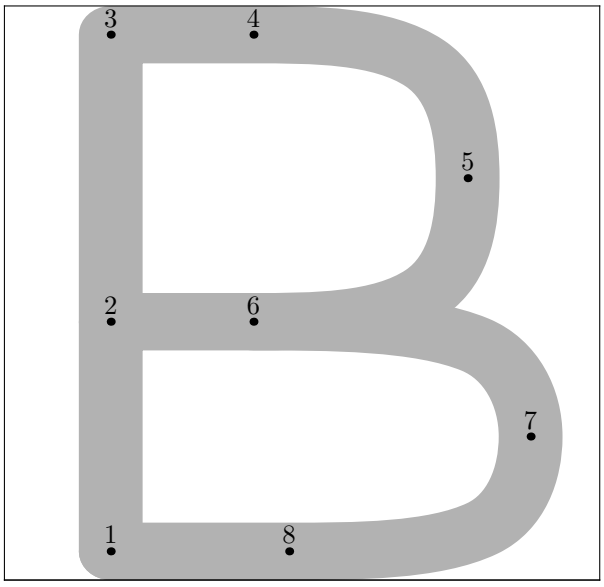
Glyph with coding number 64



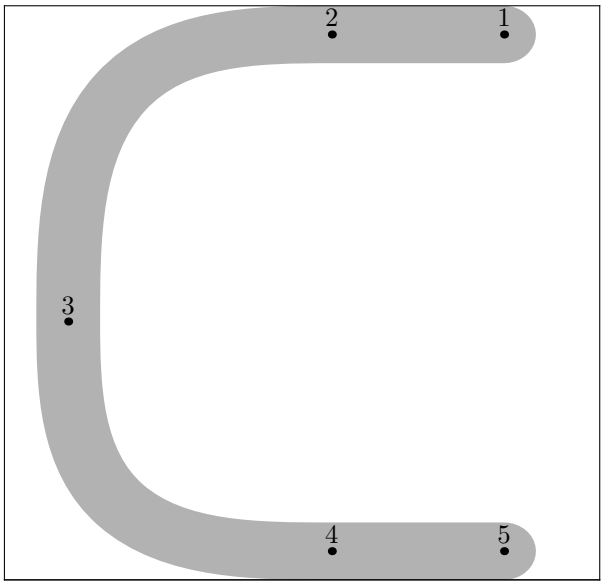
Glyph with coding number 65



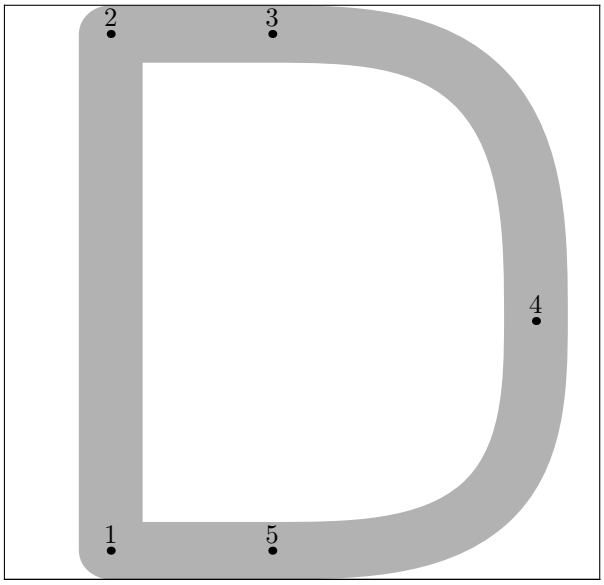
Glyph with coding number 66



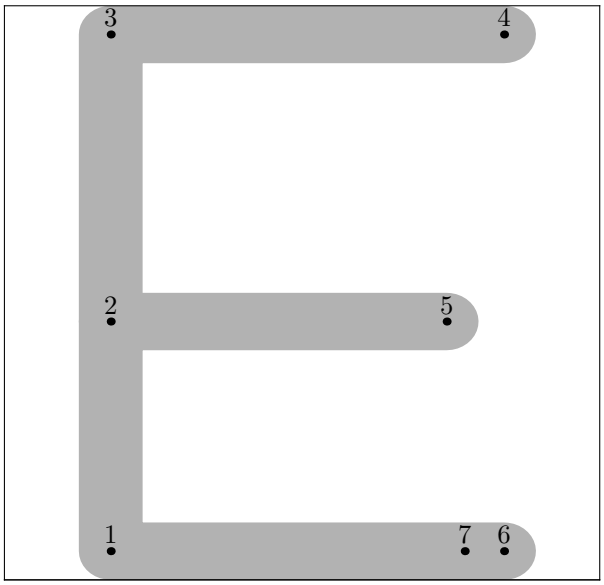
Glyph with coding number 67



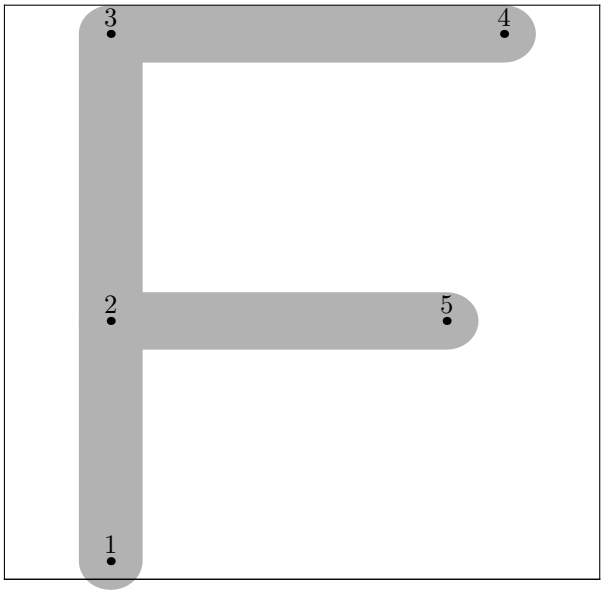
Glyph with coding number 68



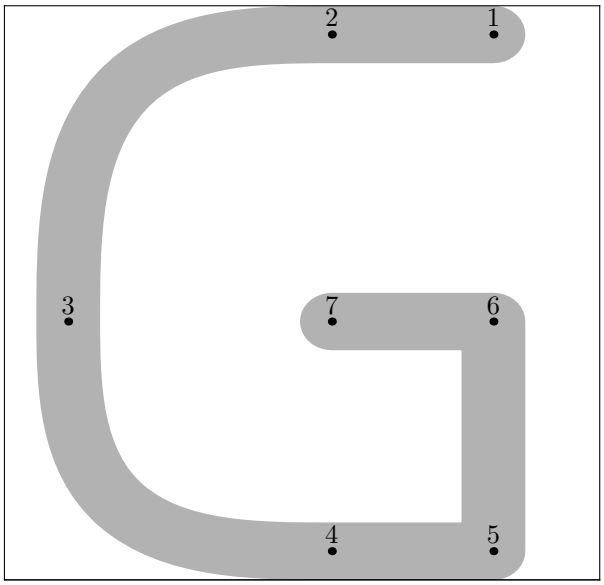
Glyph with coding number 69



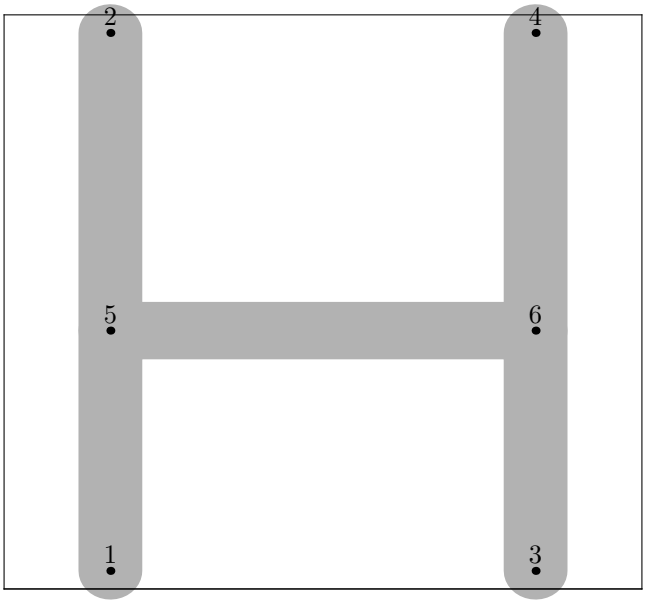
Glyph with coding number 70



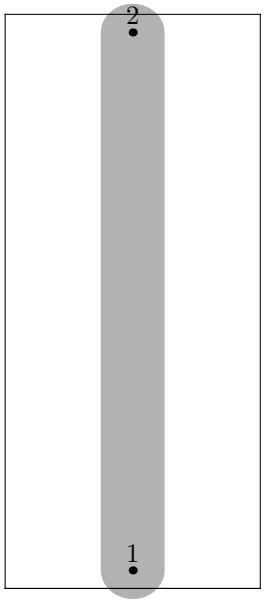
Glyph with coding number 71



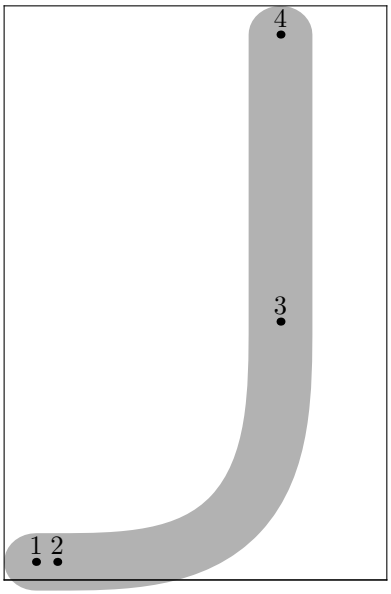
Glyph with coding number 72



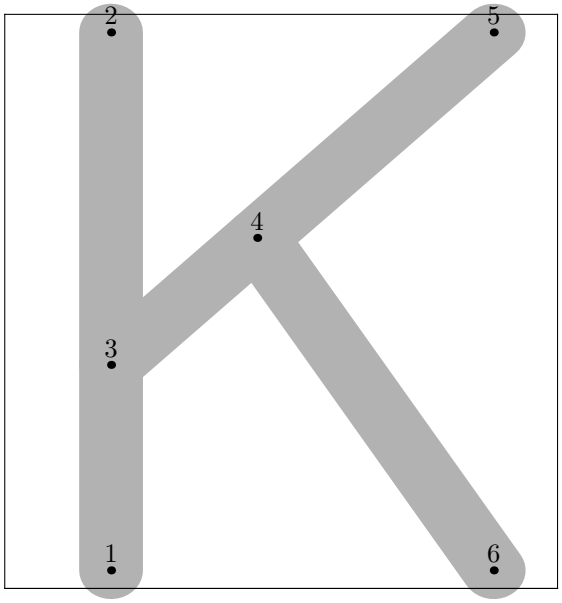
Glyph with coding number 73



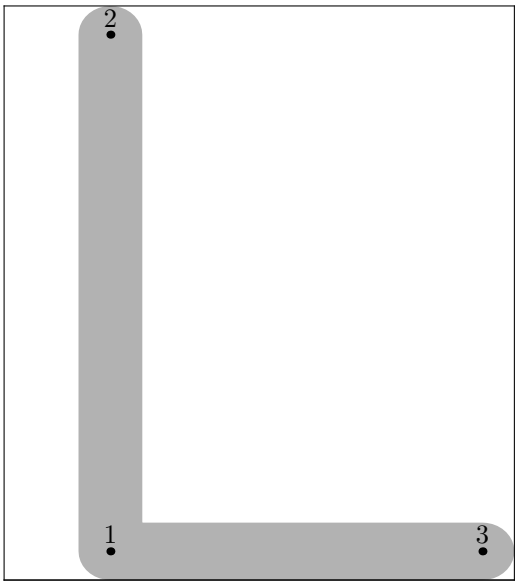
Glyph with coding number 74



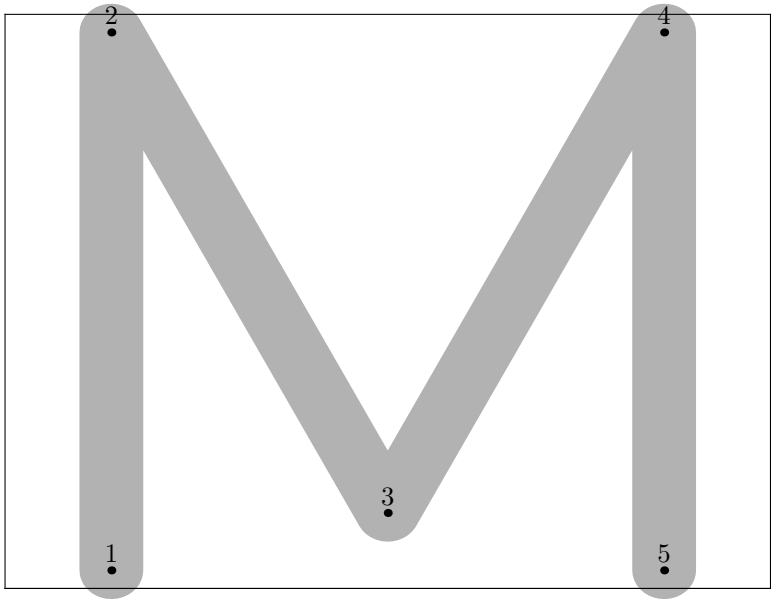
Glyph with coding number 75



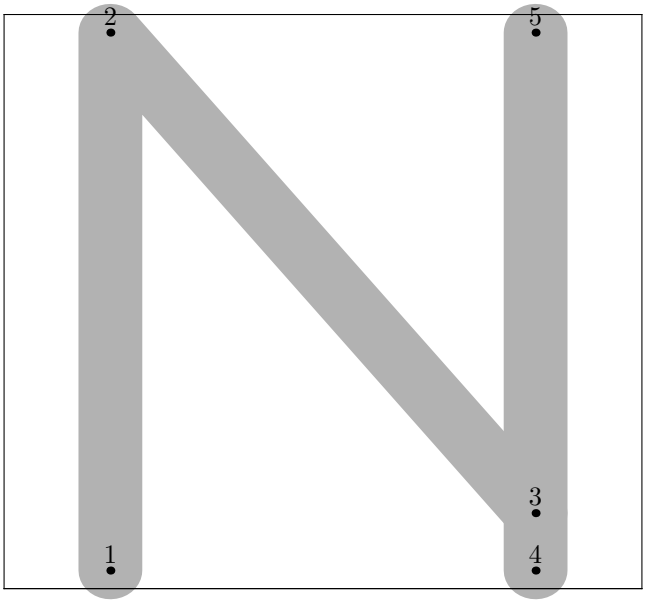
Glyph with coding number 76



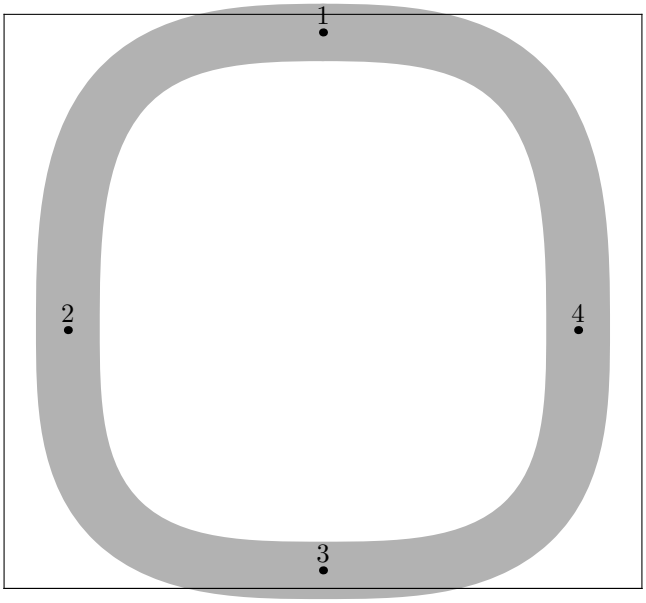
Glyph with coding number 77



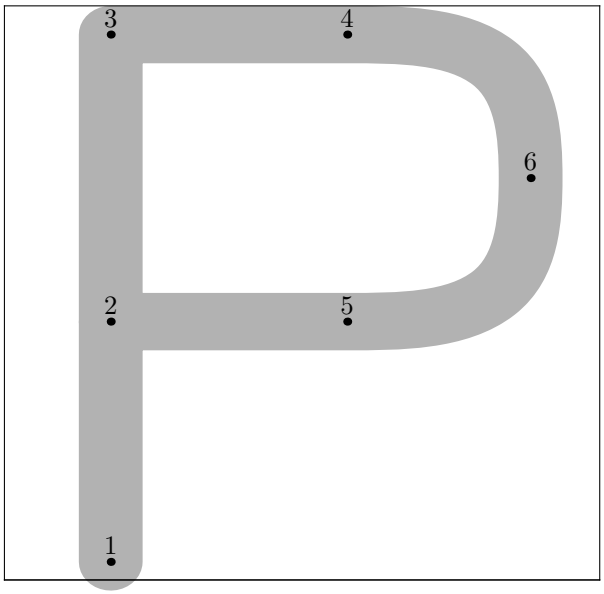
Glyph with coding number 78



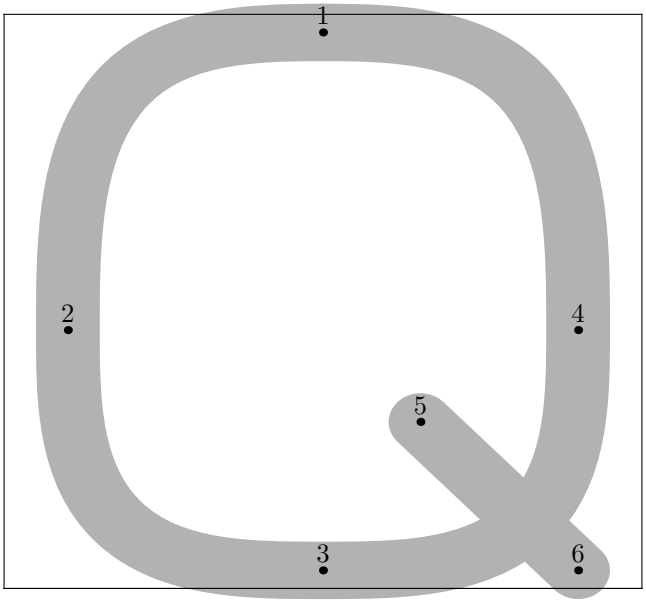
Glyph with coding number 79



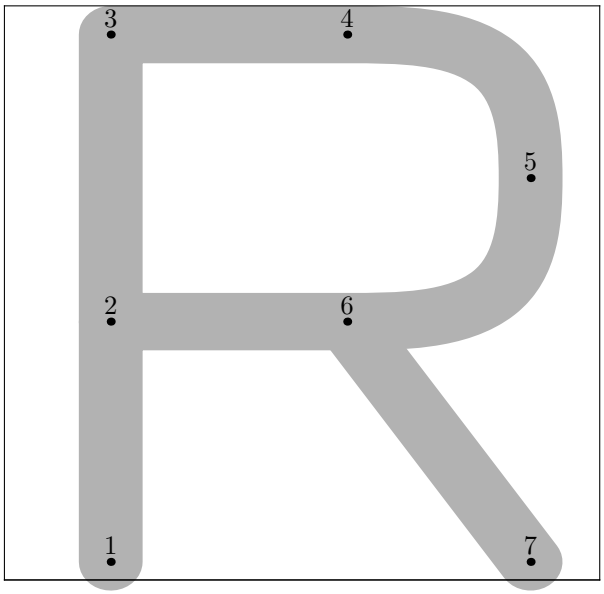
Glyph with coding number 80



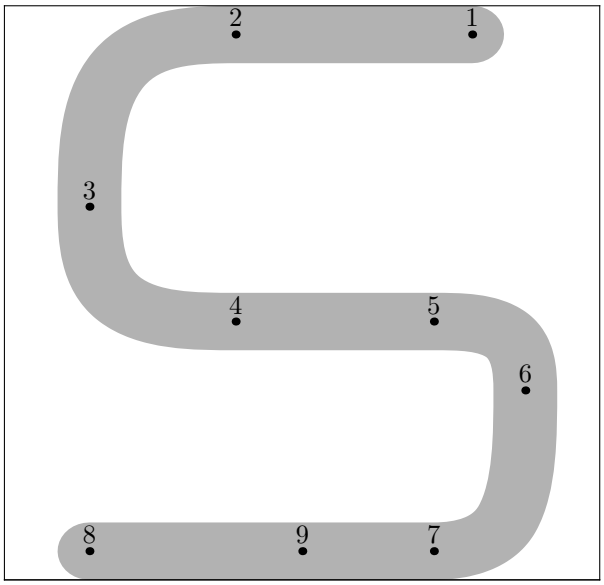
Glyph with coding number 81



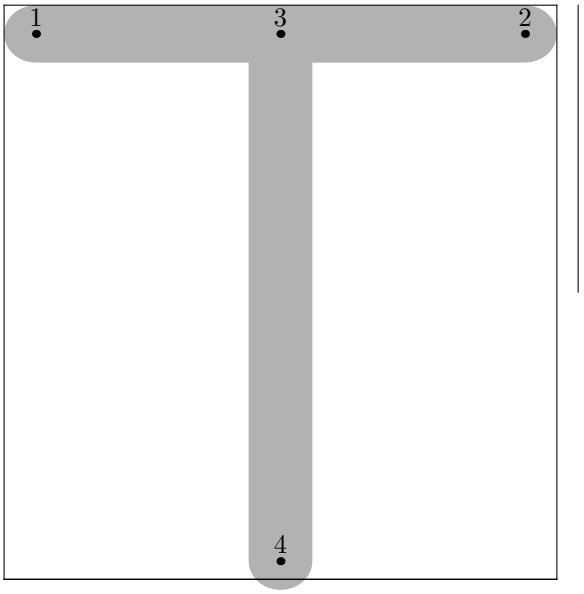
Glyph with coding number 82



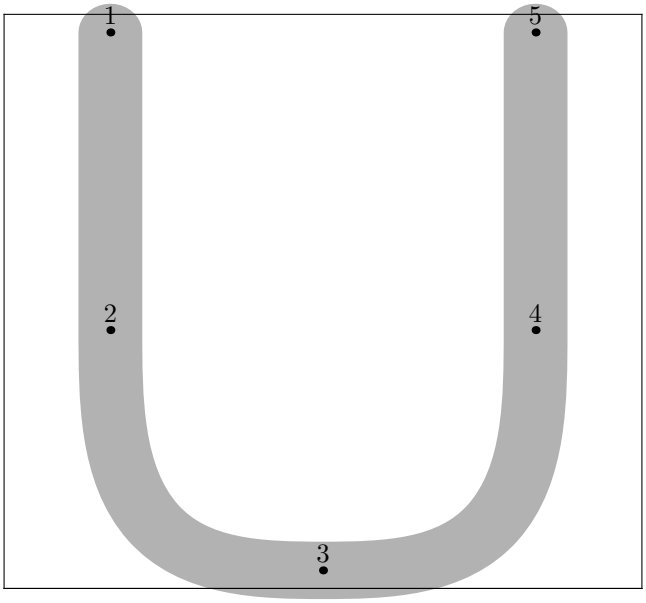
Glyph with coding number 83



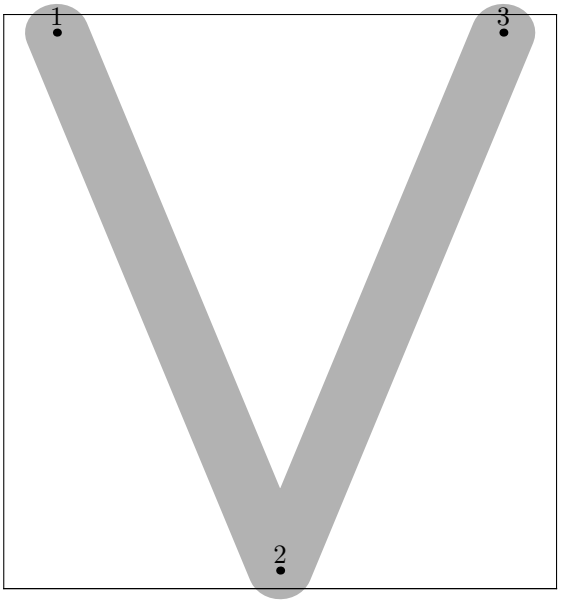
Glyph with coding number 84



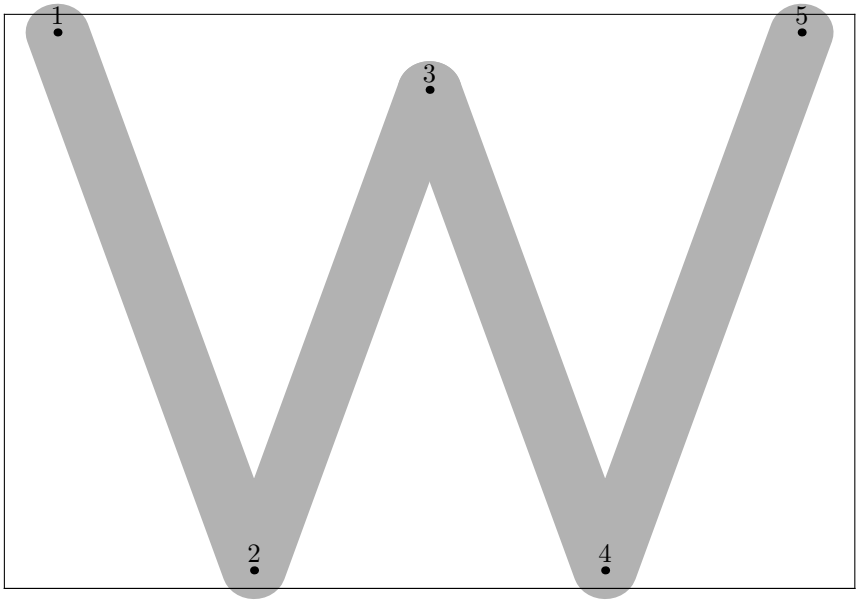
Glyph with coding number 85



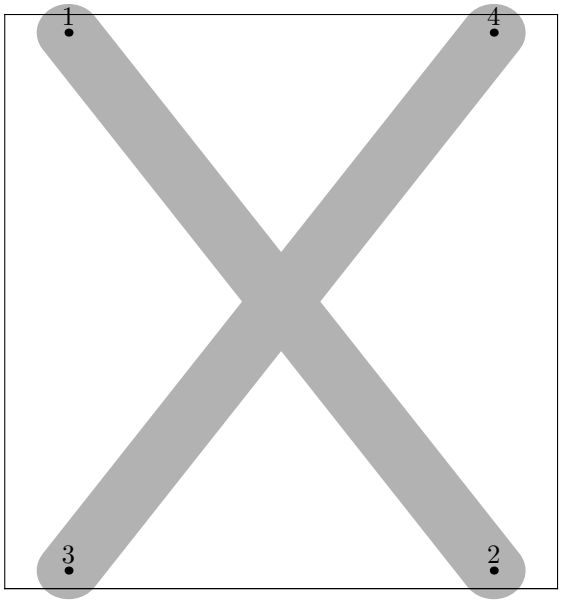
Glyph with coding number 86



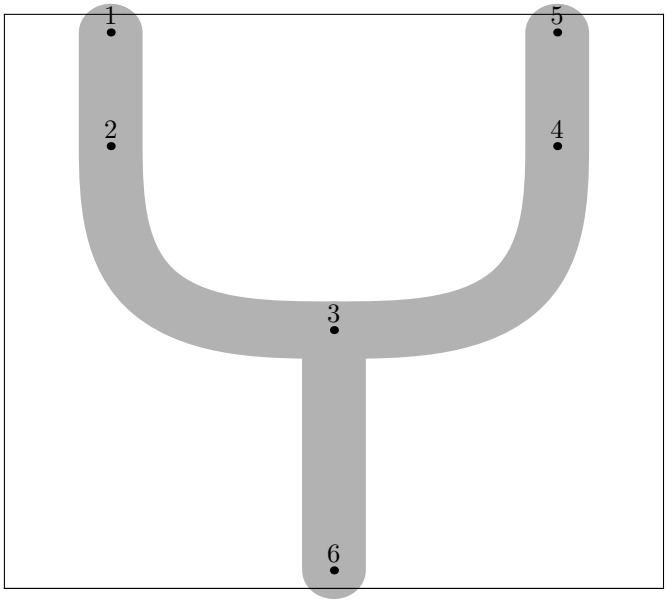
Glyph with coding number 87



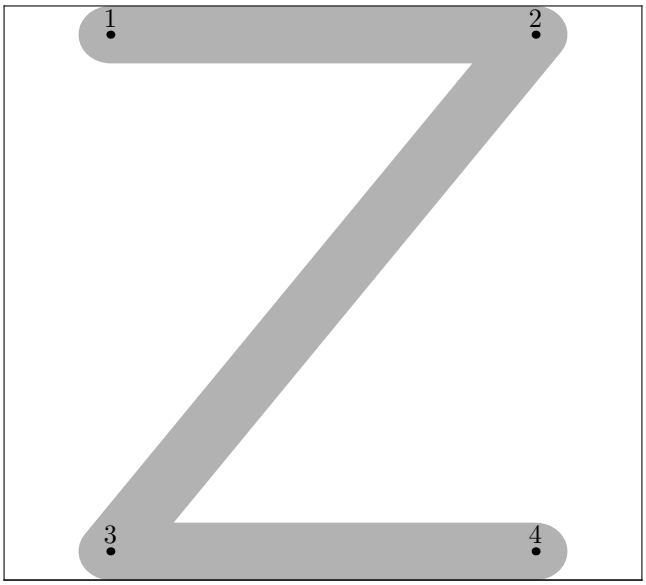
Glyph with coding number 88



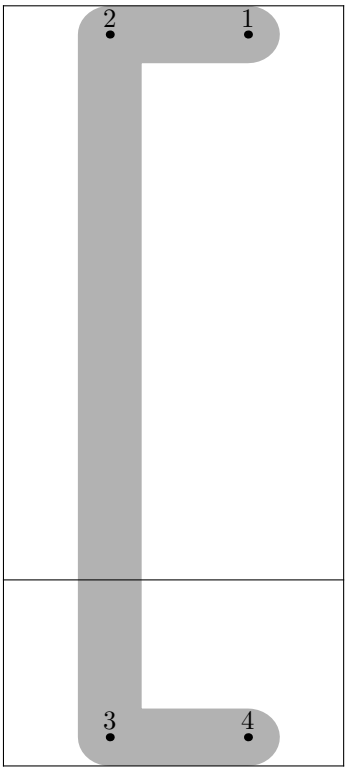
Glyph with coding number 89



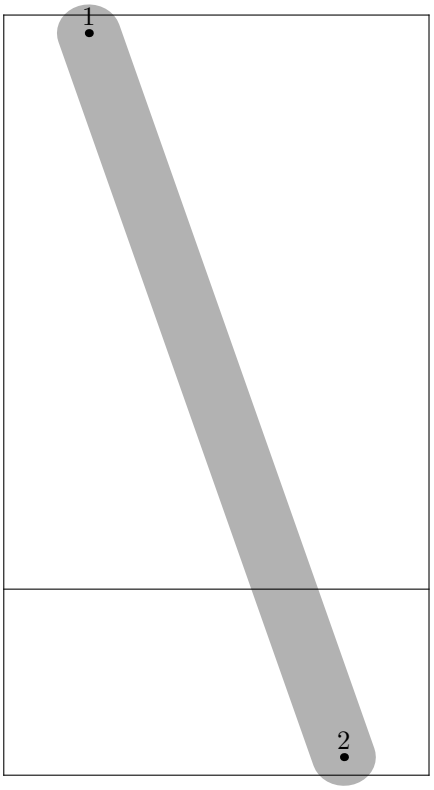
Glyph with coding number 90



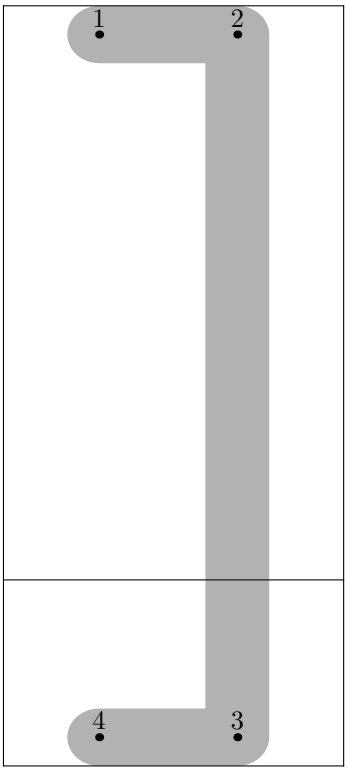
Glyph with coding number 91



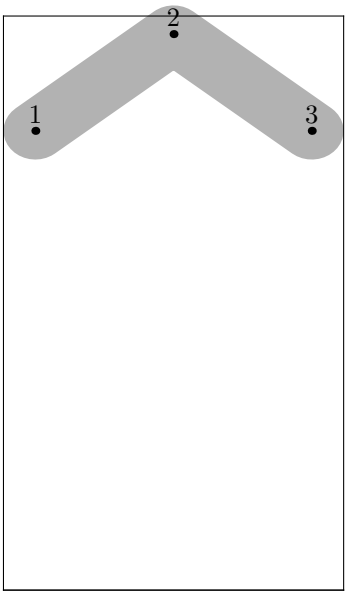
Glyph with coding number 92



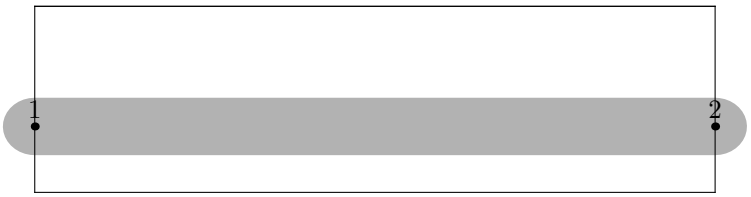
Glyph with coding number 93



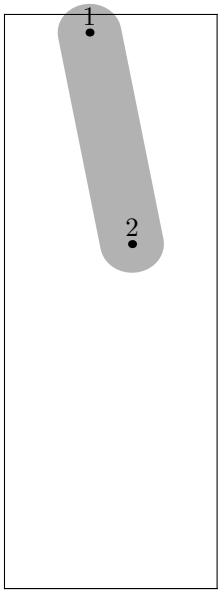
Glyph with coding number 94



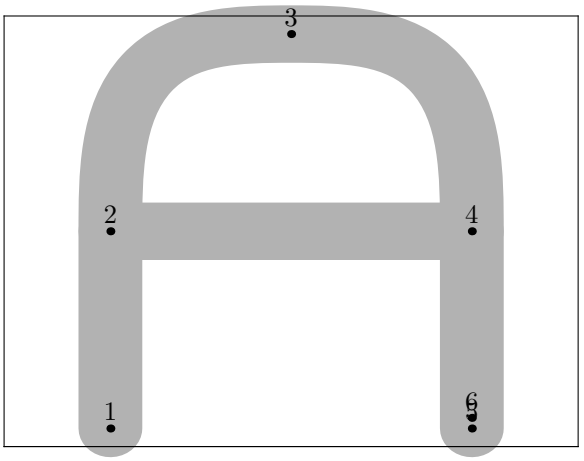
Glyph with coding number 95



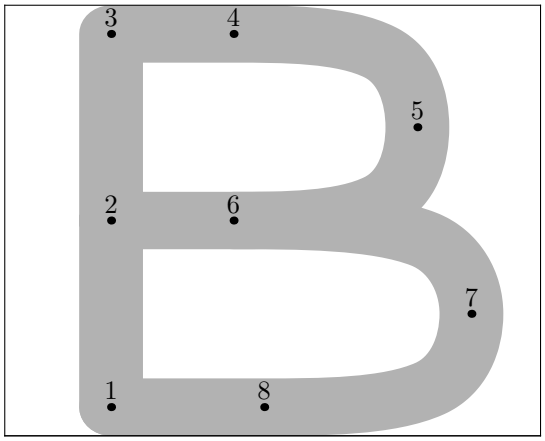
Glyph with coding number 96



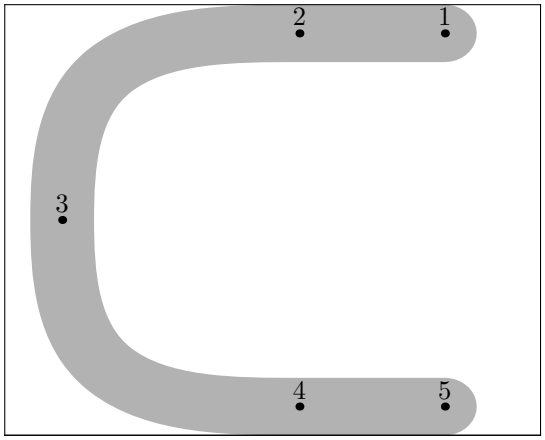
Glyph with coding number 97



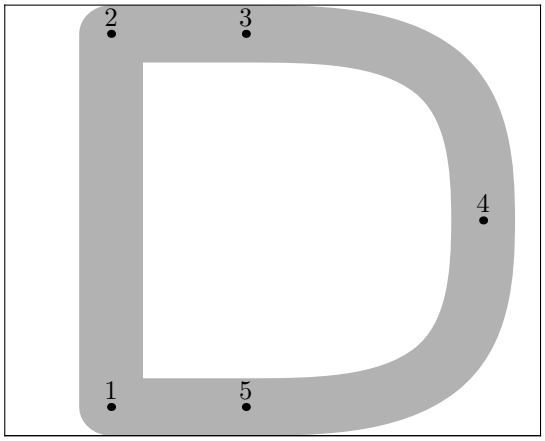
Glyph with coding number 98



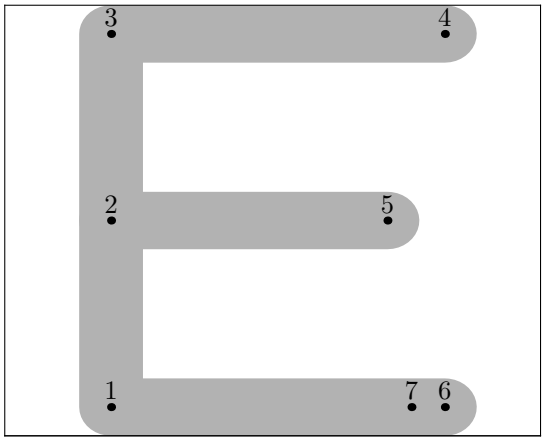
Glyph with coding number 99



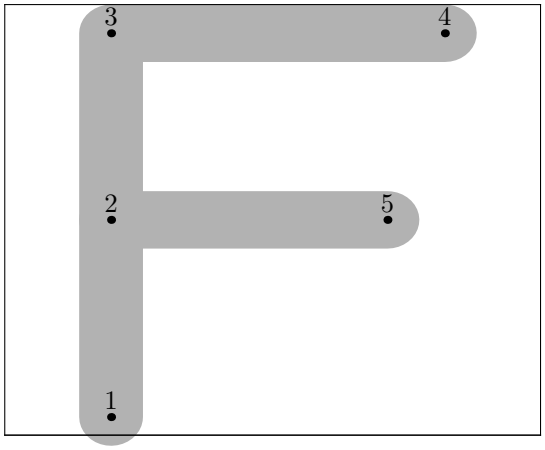
Glyph with coding number 100



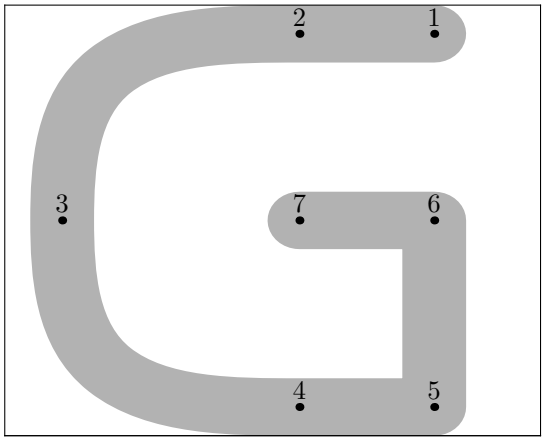
Glyph with coding number 101



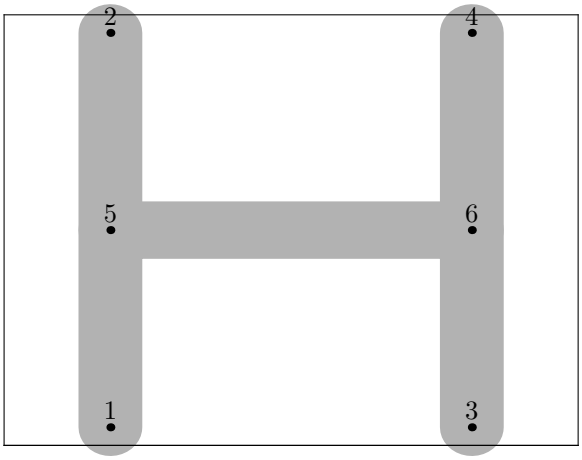
Glyph with coding number 102



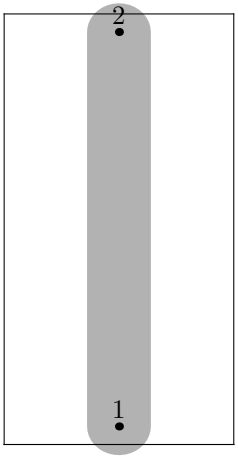
Glyph with coding number 103



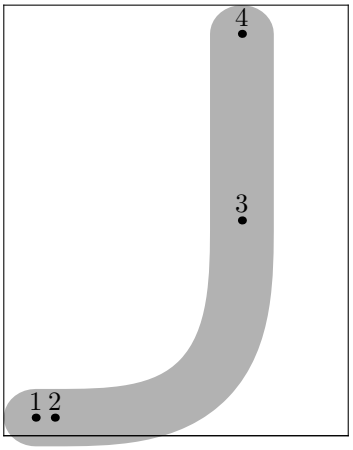
Glyph with coding number 104



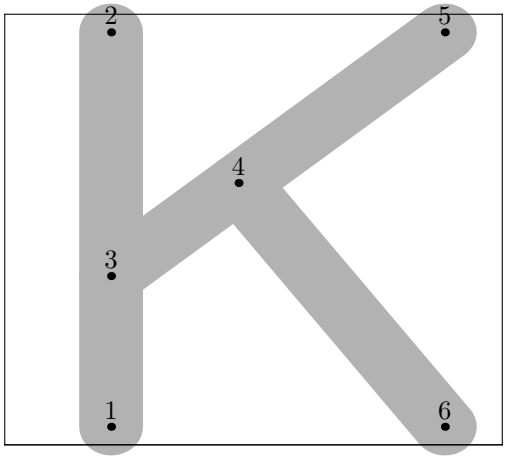
Glyph with coding number 105



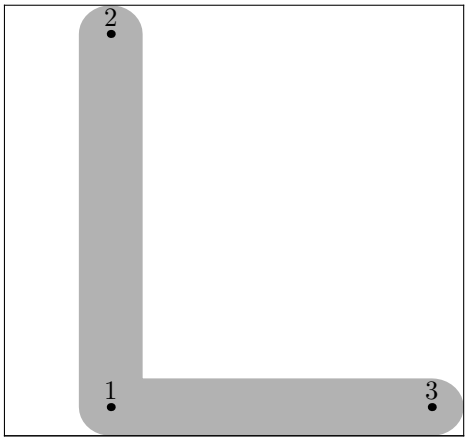
Glyph with coding number 106



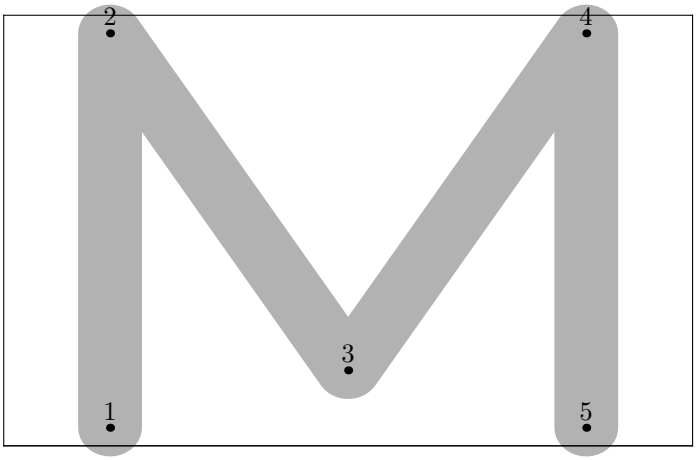
Glyph with coding number 107



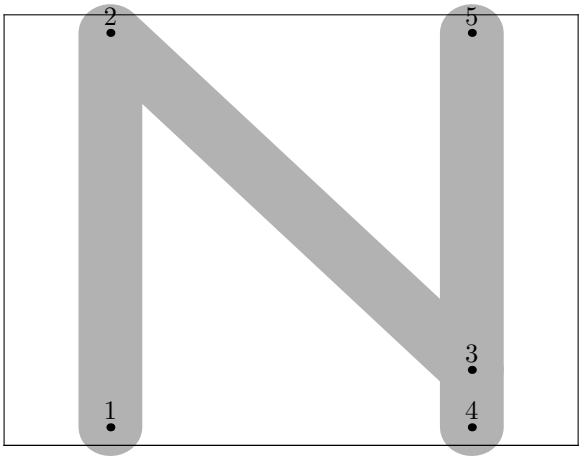
Glyph with coding number 108



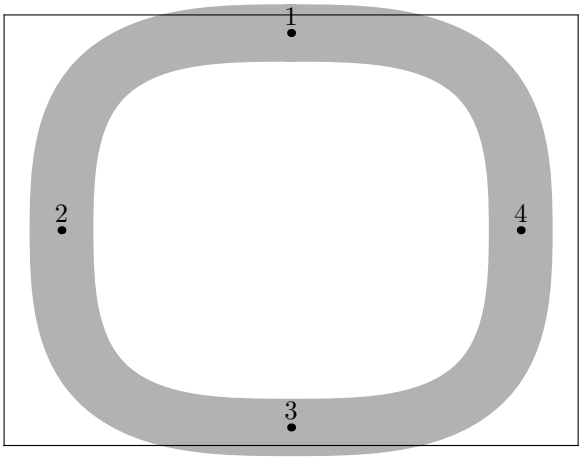
Glyph with coding number 109



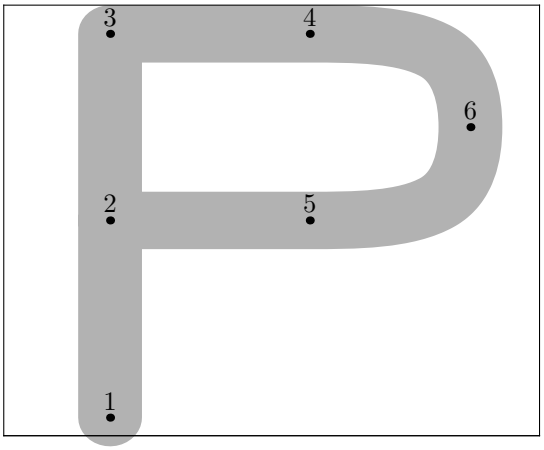
Glyph with coding number 110



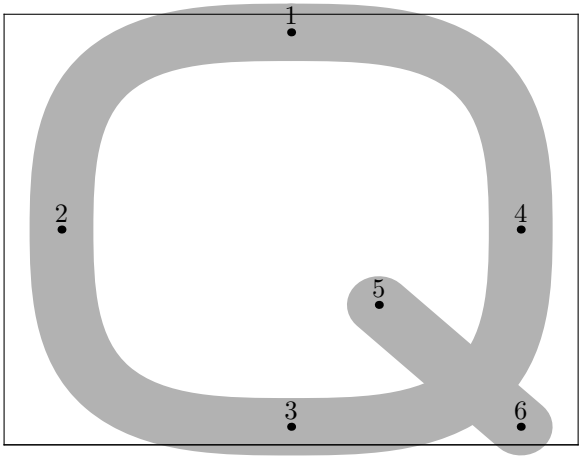
Glyph with coding number 111



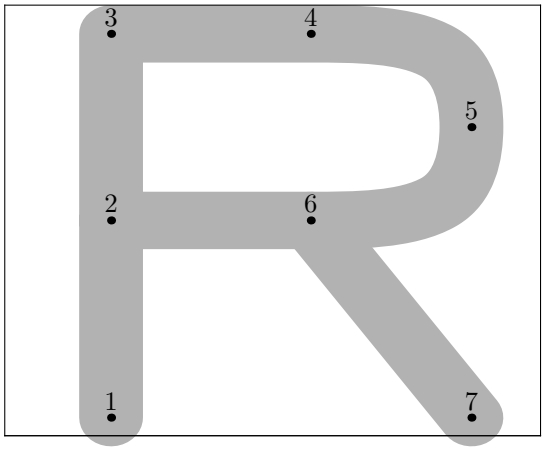
Glyph with coding number 112



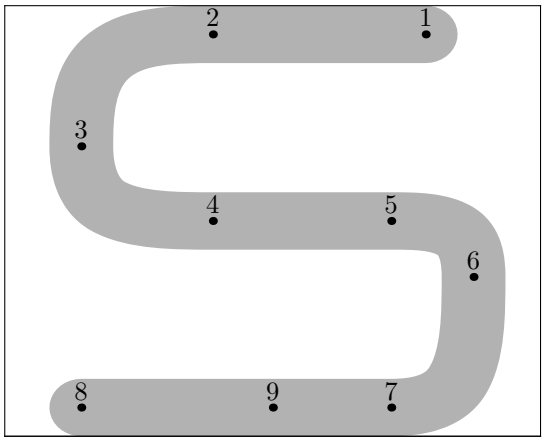
Glyph with coding number 113



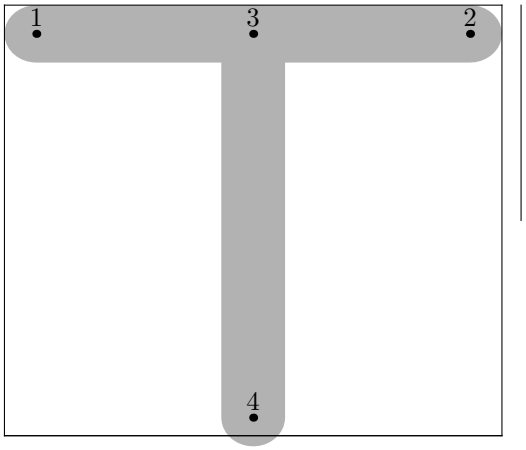
Glyph with coding number 114



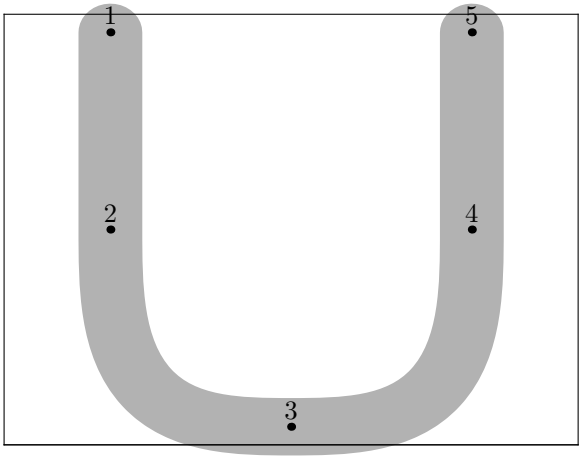
Glyph with coding number 115



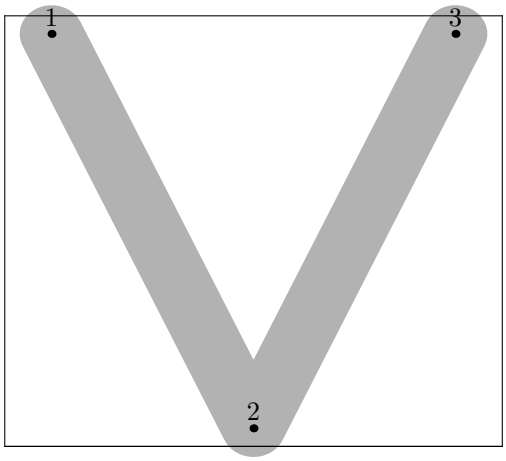
Glyph with coding number 116



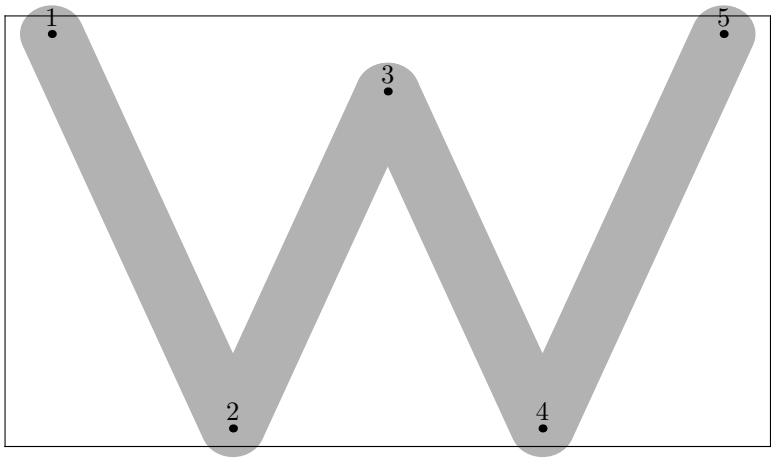
Glyph with coding number 117



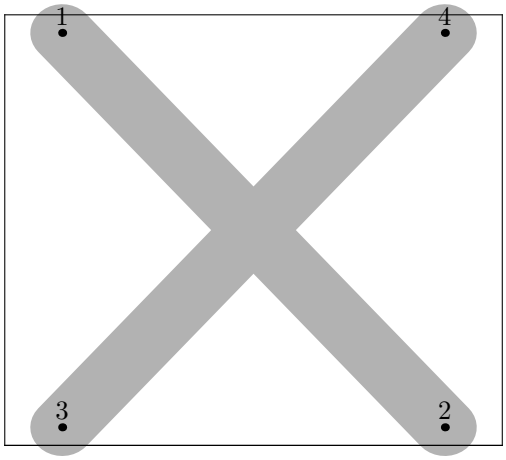
Glyph with coding number 118



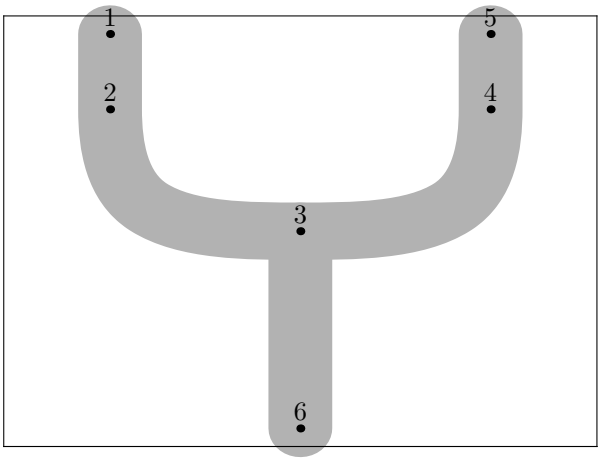
Glyph with coding number 119



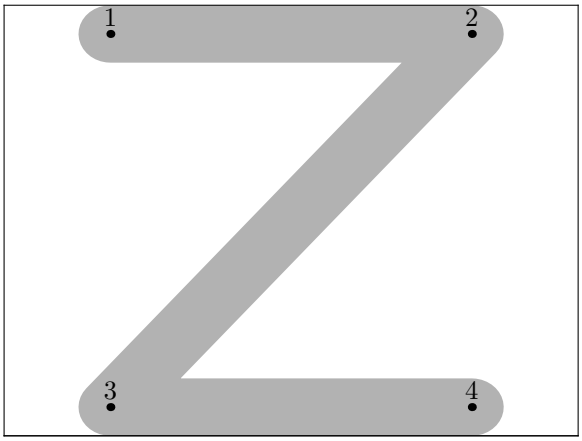
Glyph with coding number 120



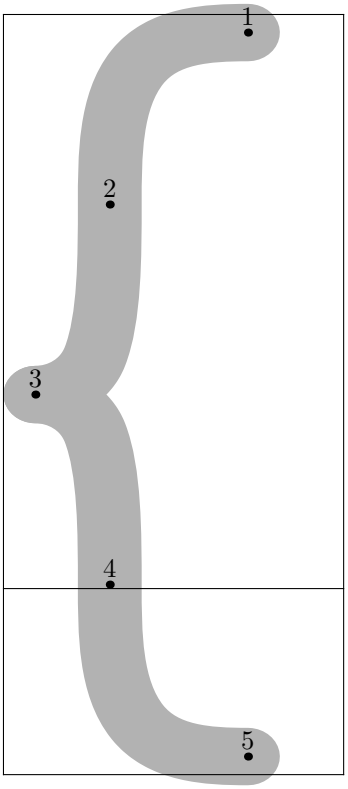
Glyph with coding number 121



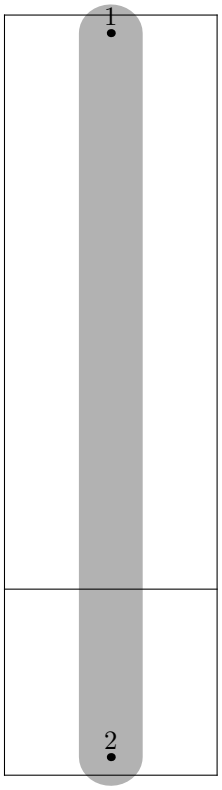
Glyph with coding number 122



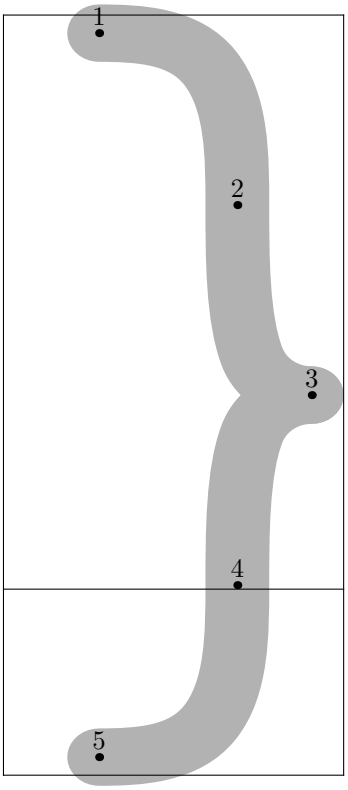
Glyph with coding number 123



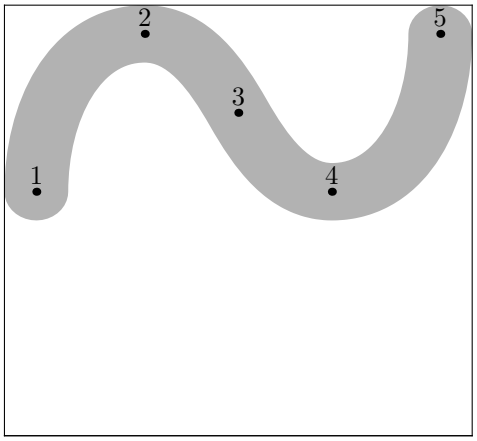
Glyph with coding number 124



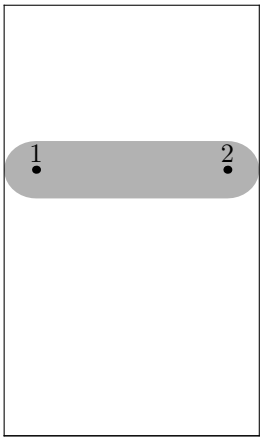
Glyph with coding number 125



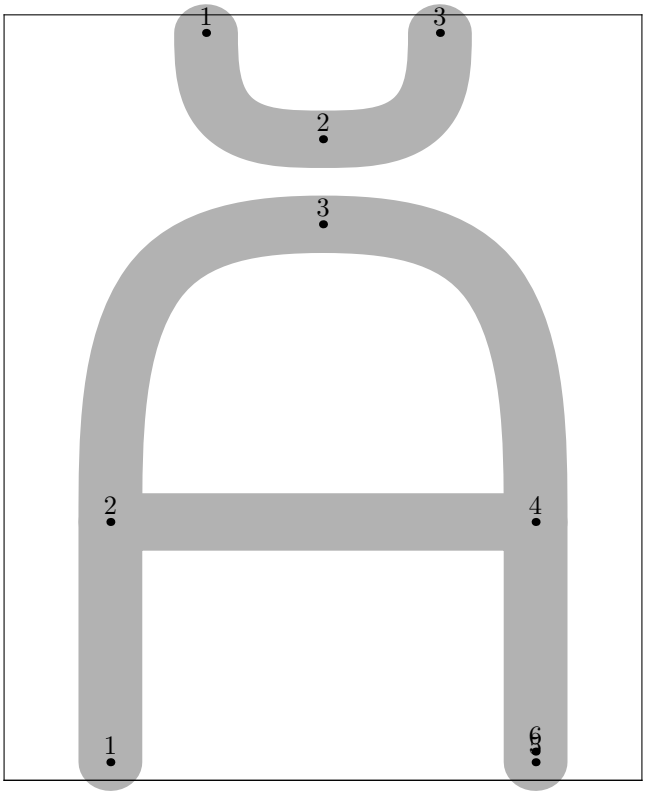
Glyph with coding number 126



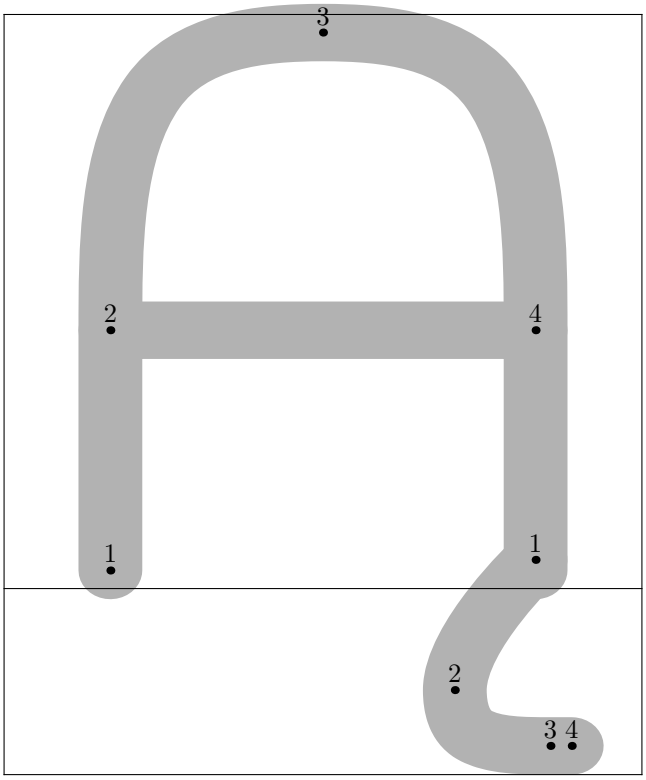
Glyph with coding number 127



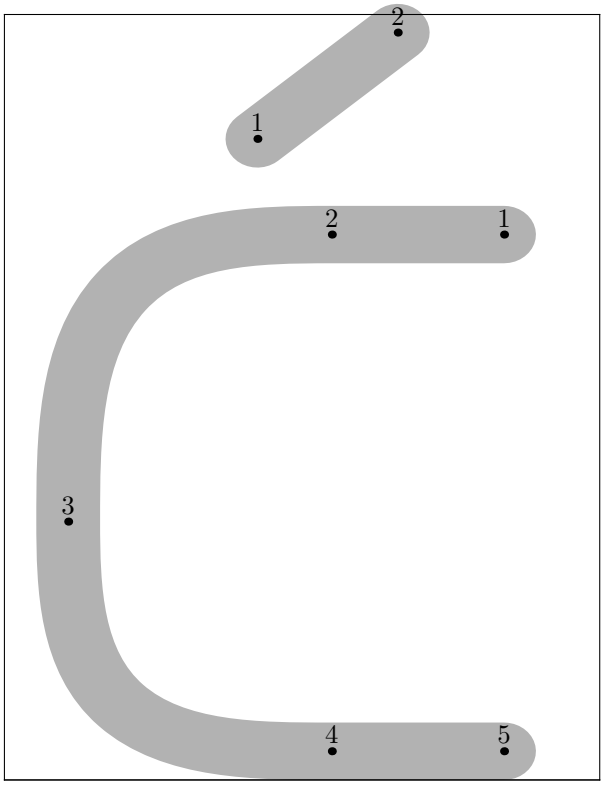
Glyph with coding number 128



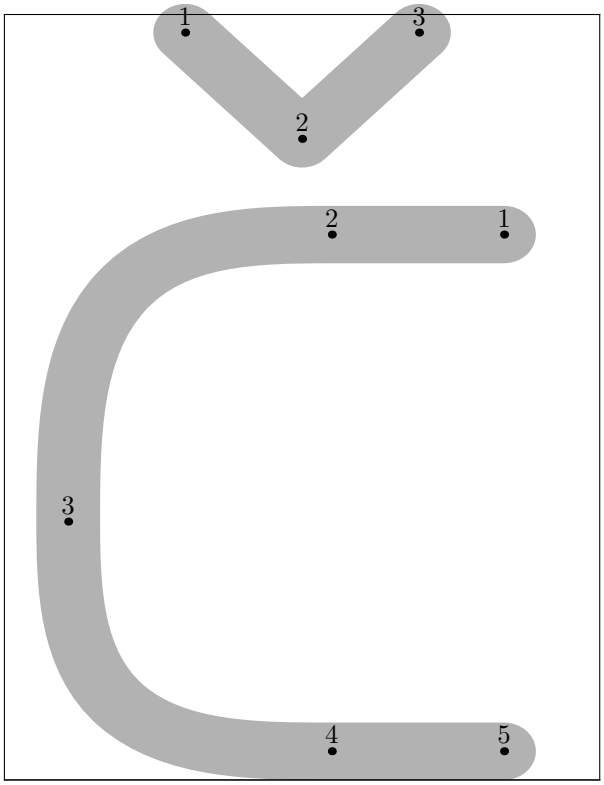
Glyph with coding number 129



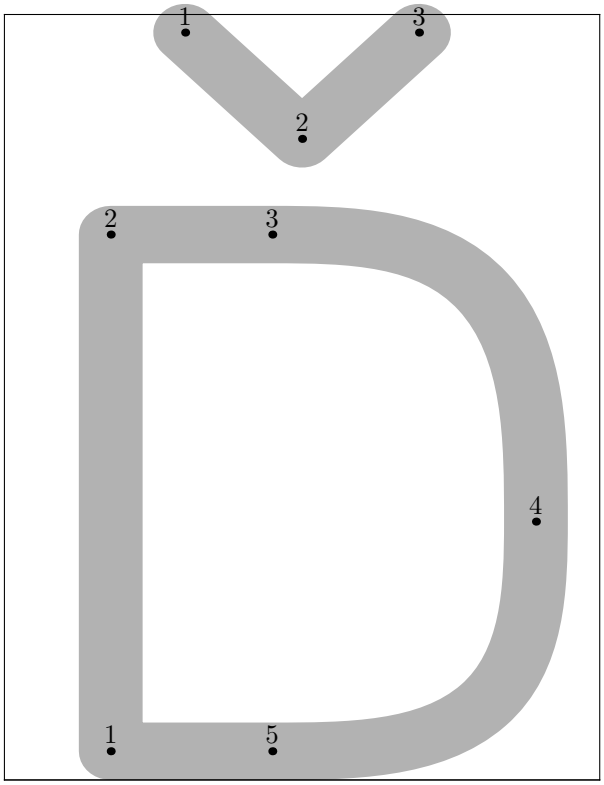
Glyph with coding number 130



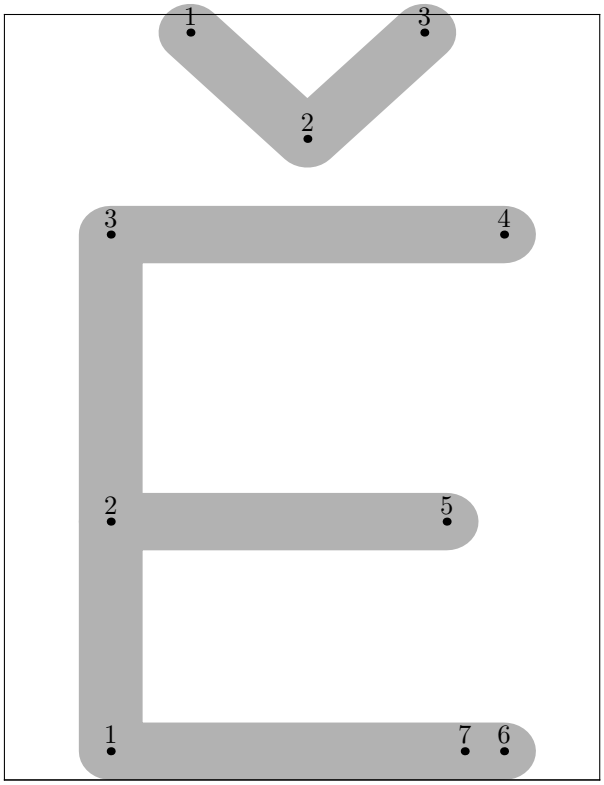
Glyph with coding number 131



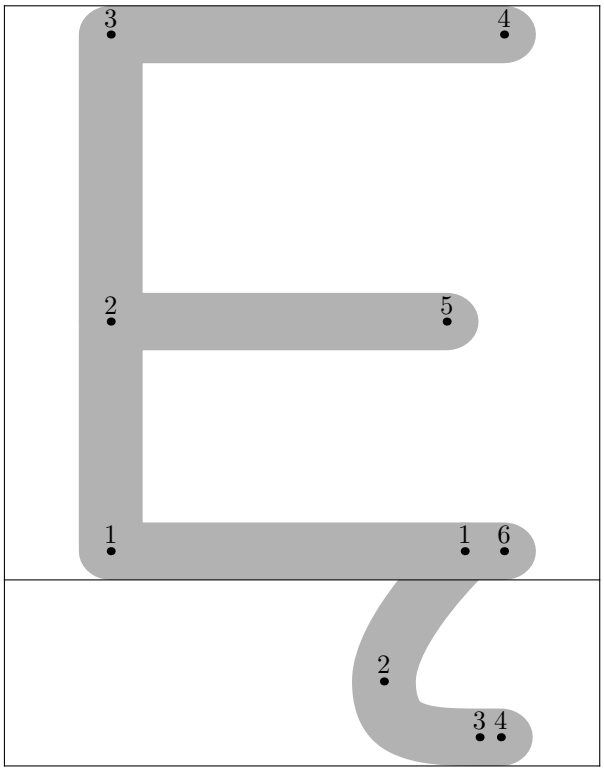
Glyph with coding number 132



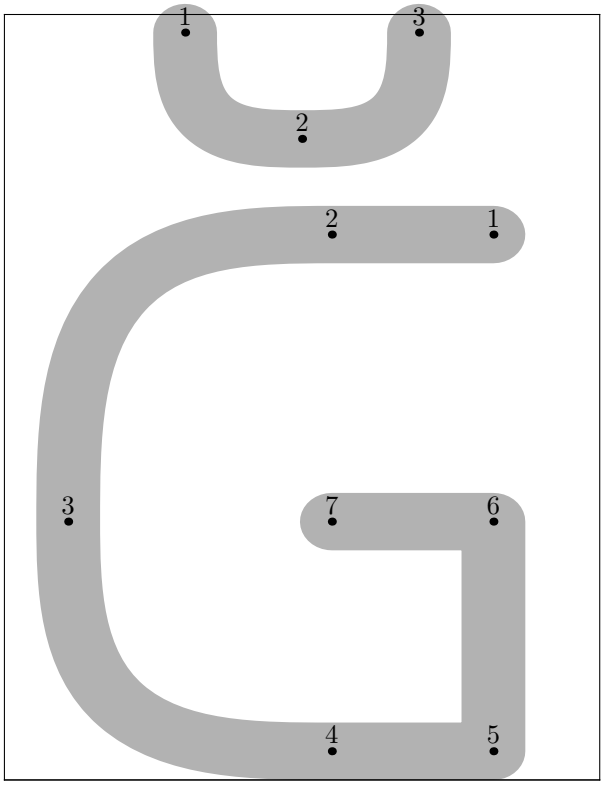
Glyph with coding number 133



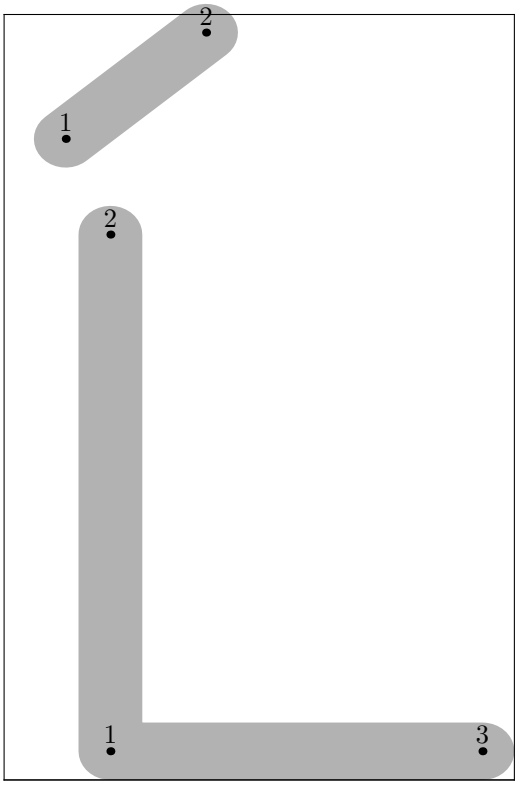
Glyph with coding number 134



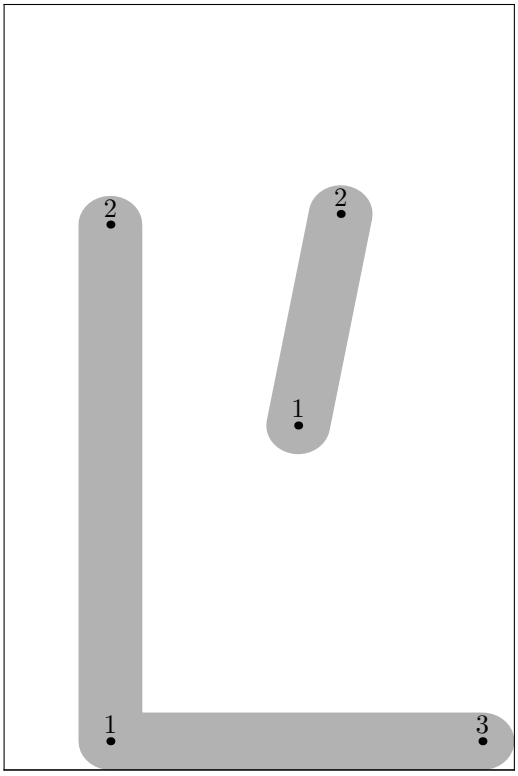
Glyph with coding number 135



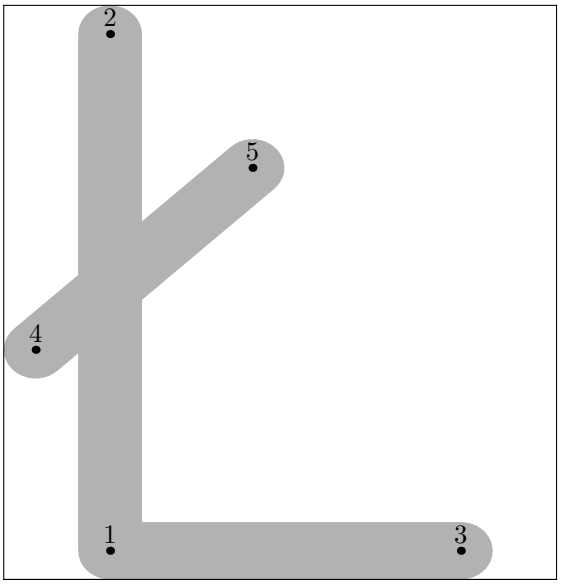
Glyph with coding number 136



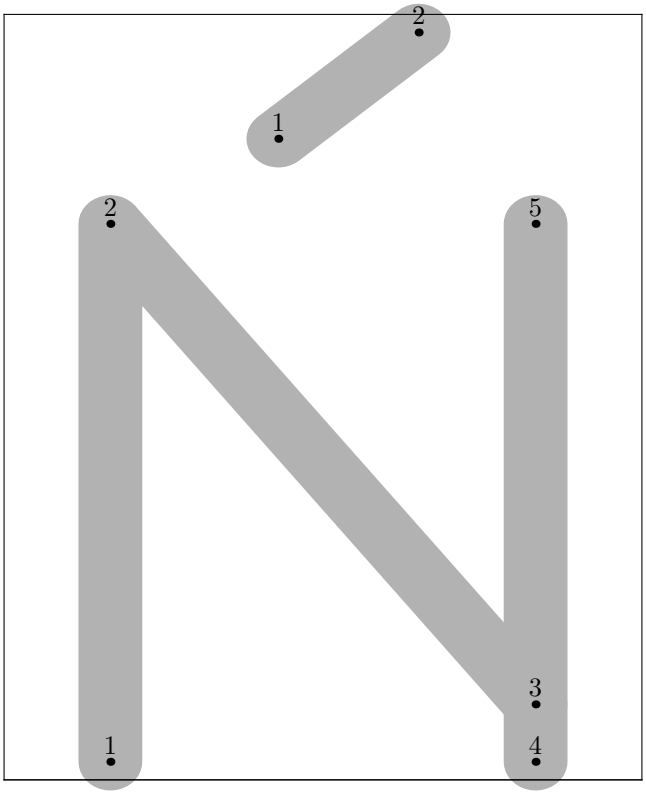
Glyph with coding number 137



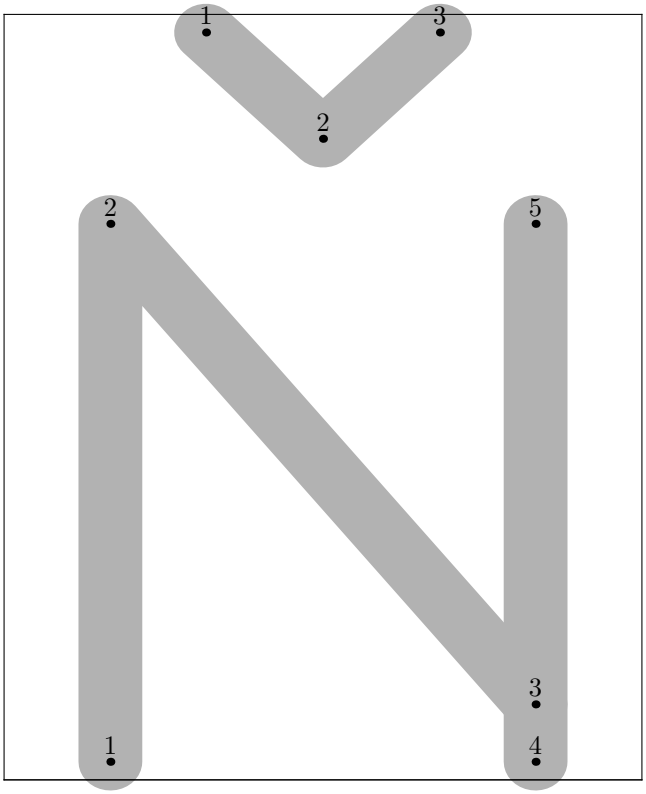
Glyph with coding number 138



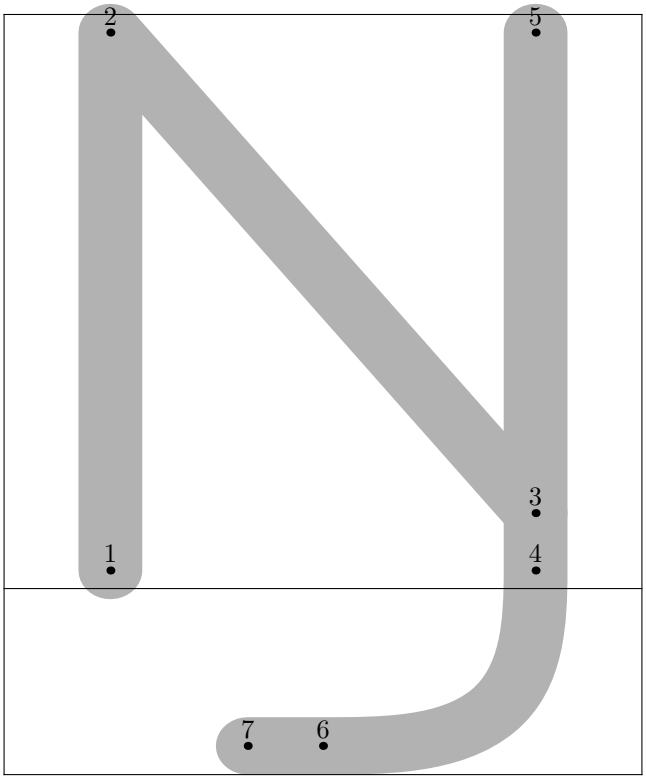
Glyph with coding number 139



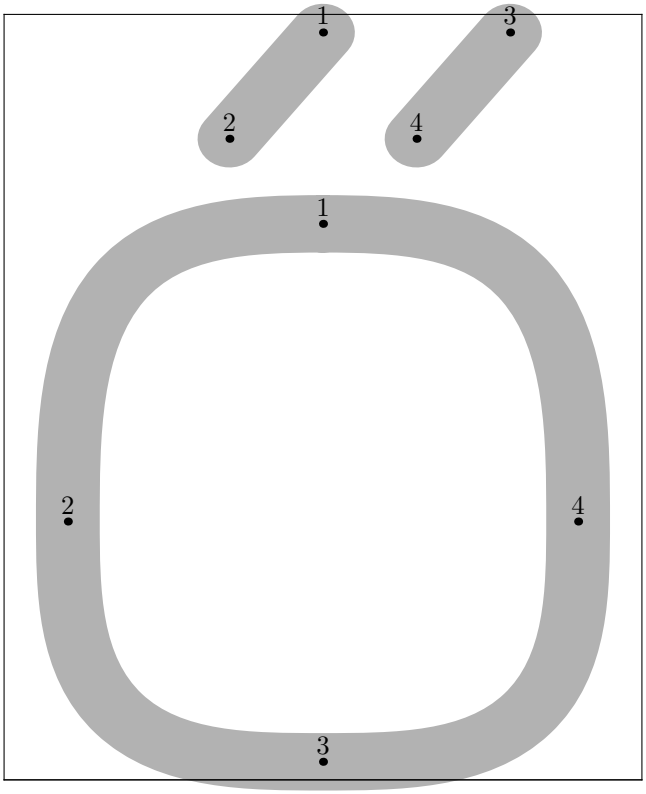
Glyph with coding number 140



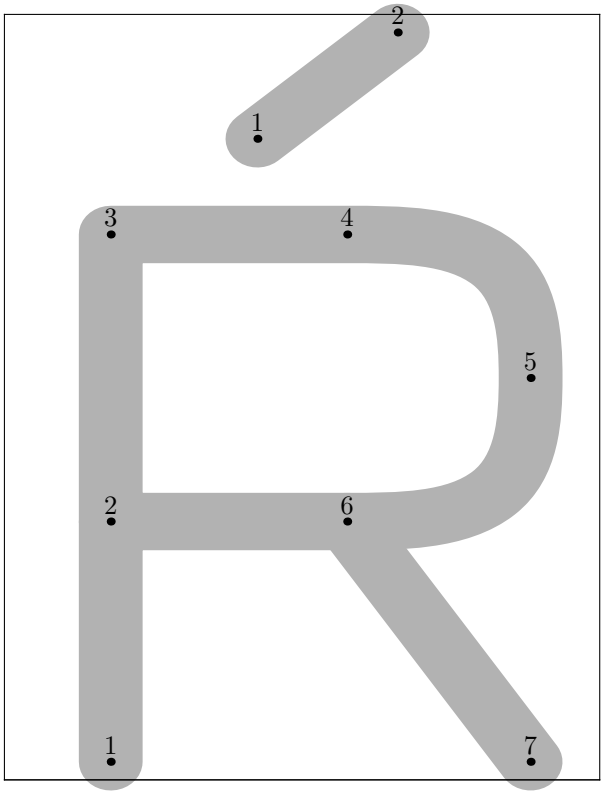
Glyph with coding number 141



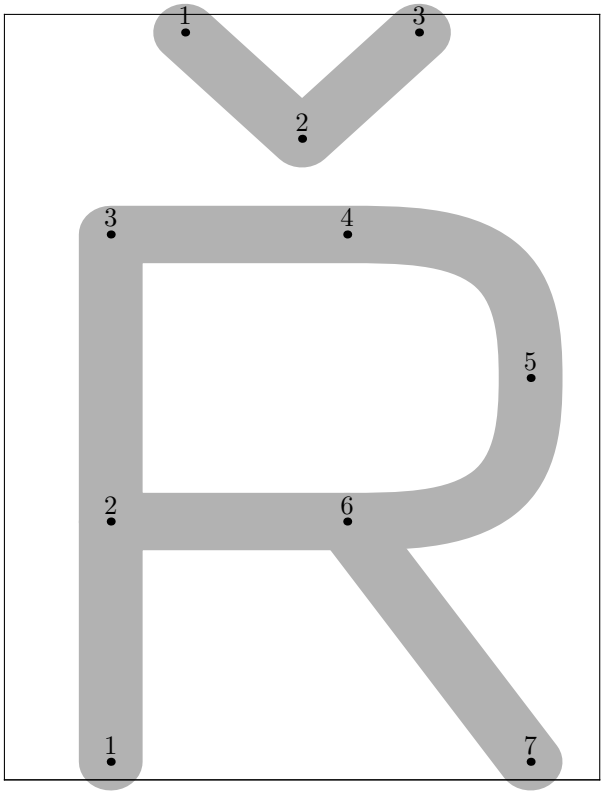
Glyph with coding number 142



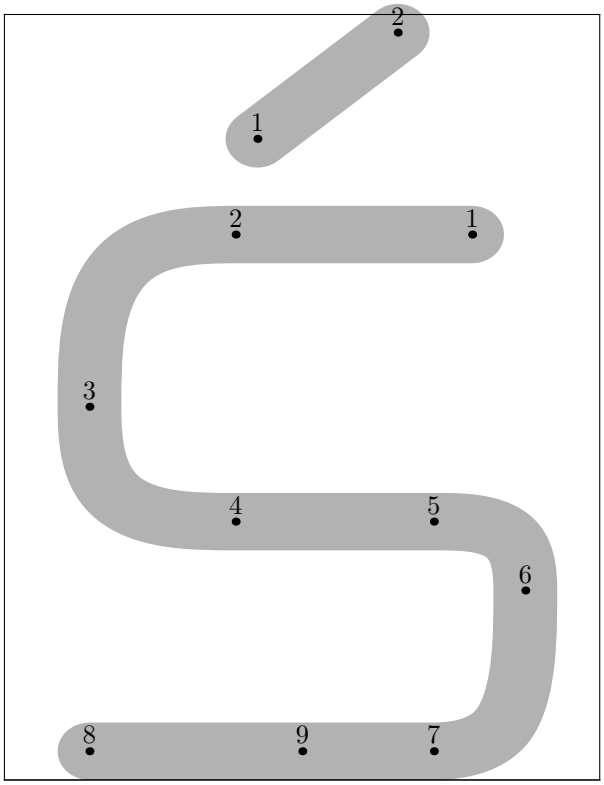
Glyph with coding number 143



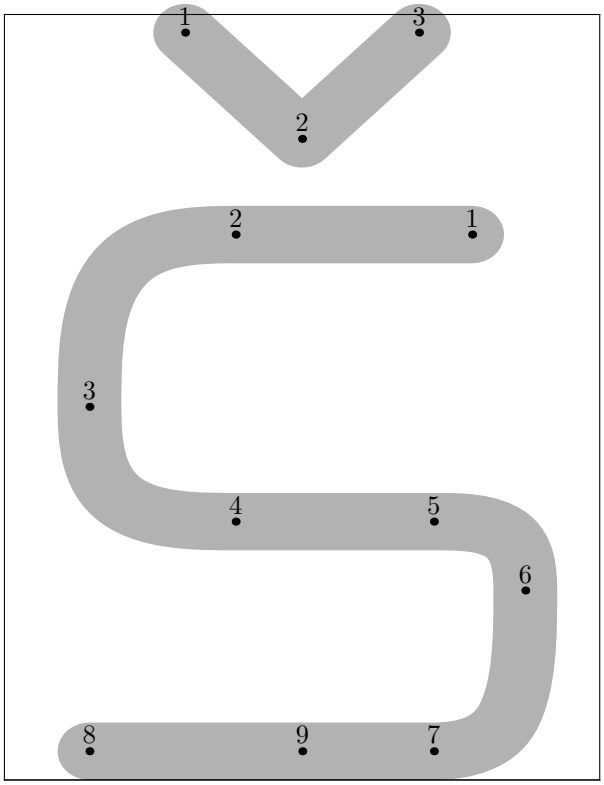
Glyph with coding number 144



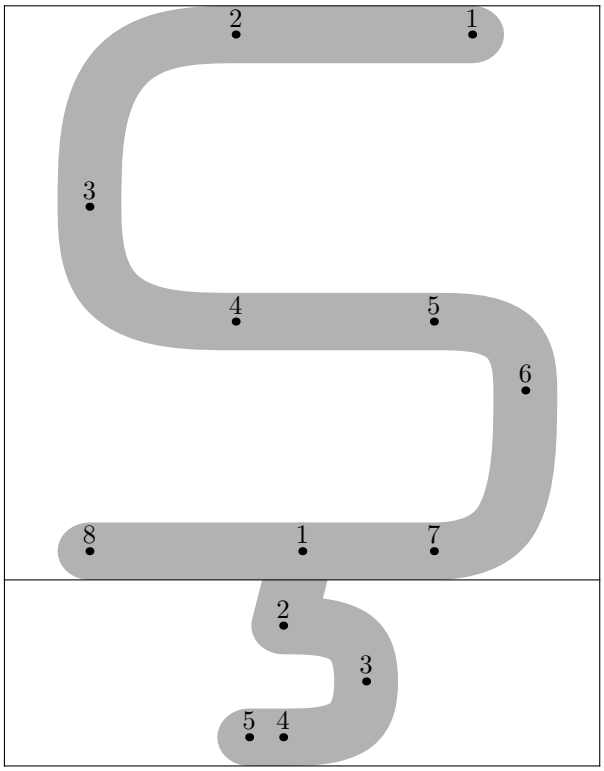
Glyph with coding number 145



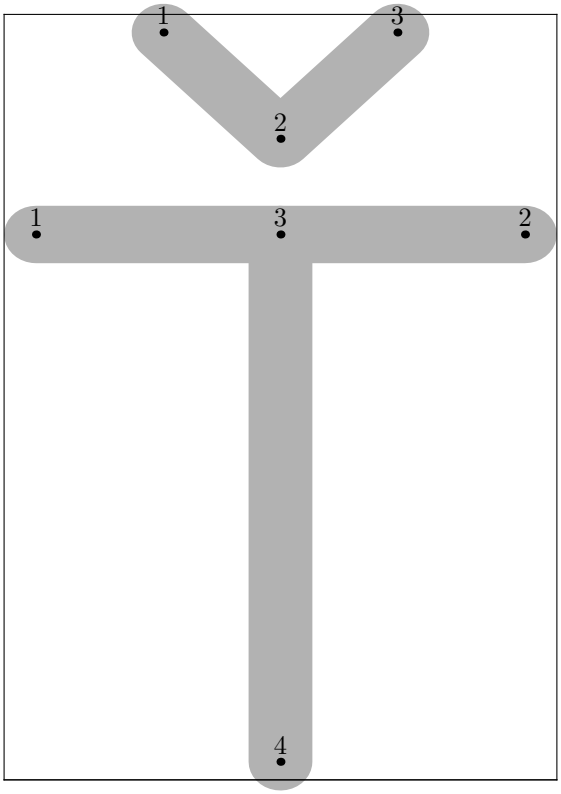
Glyph with coding number 146



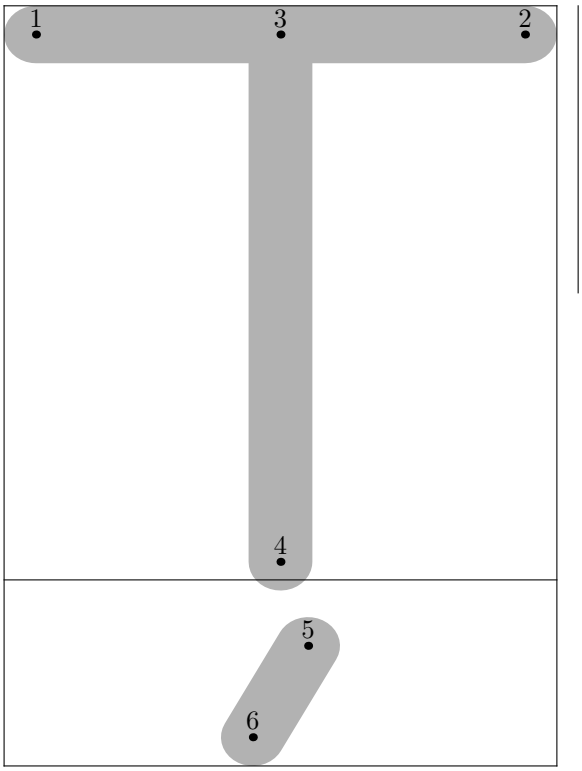
Glyph with coding number 147



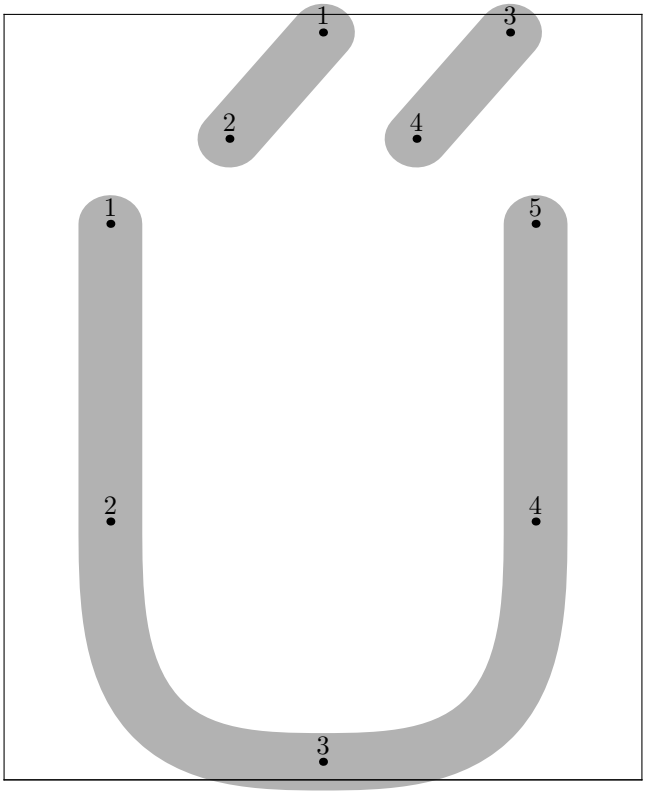
Glyph with coding number 148



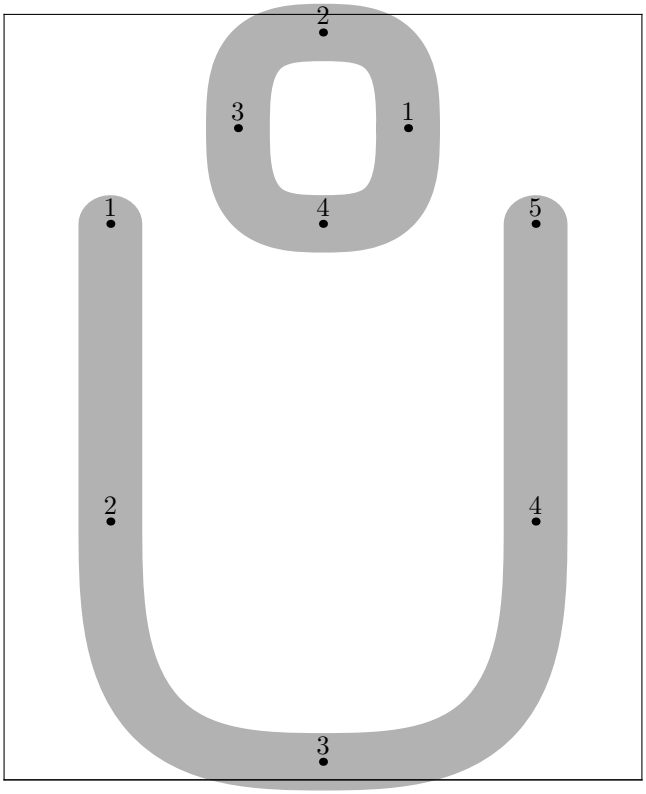
Glyph with coding number 149



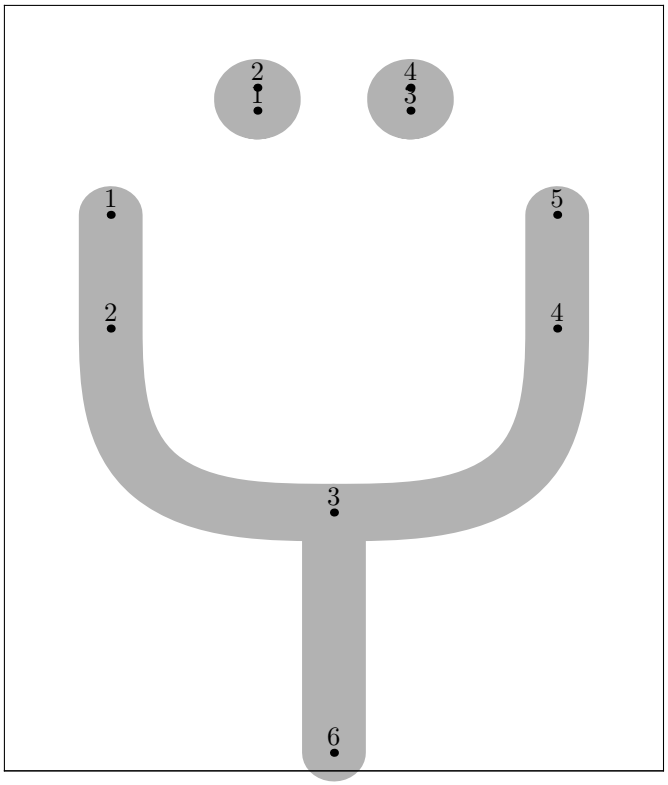
Glyph with coding number 150



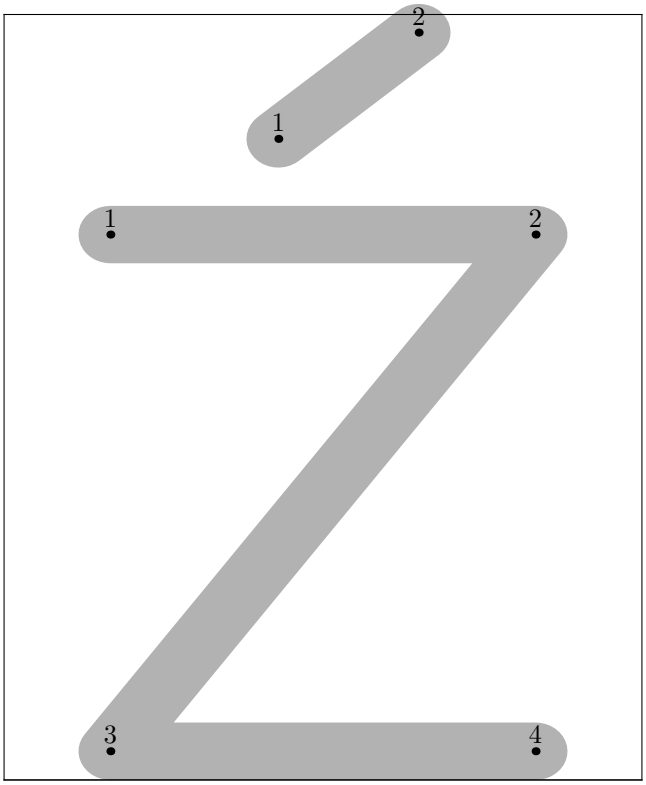
Glyph with coding number 151



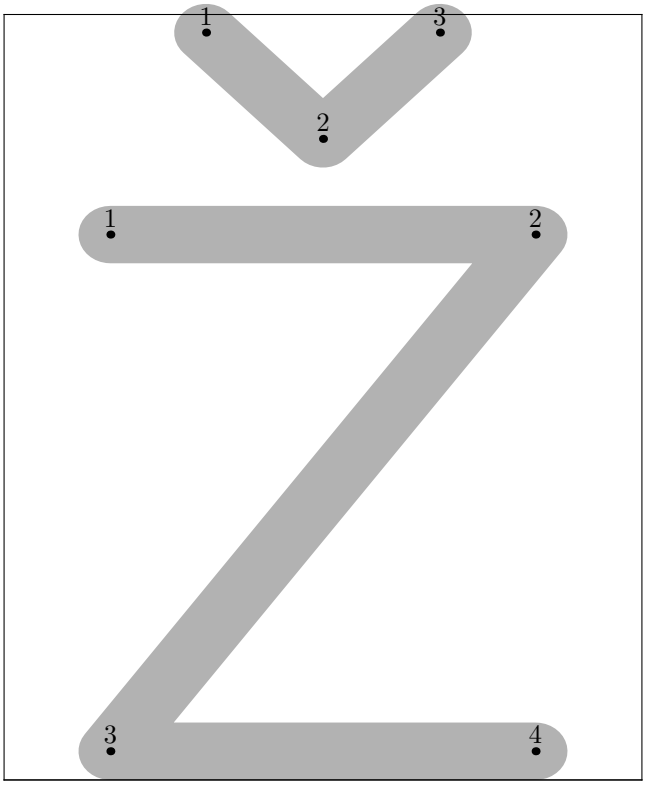
Glyph with coding number 152



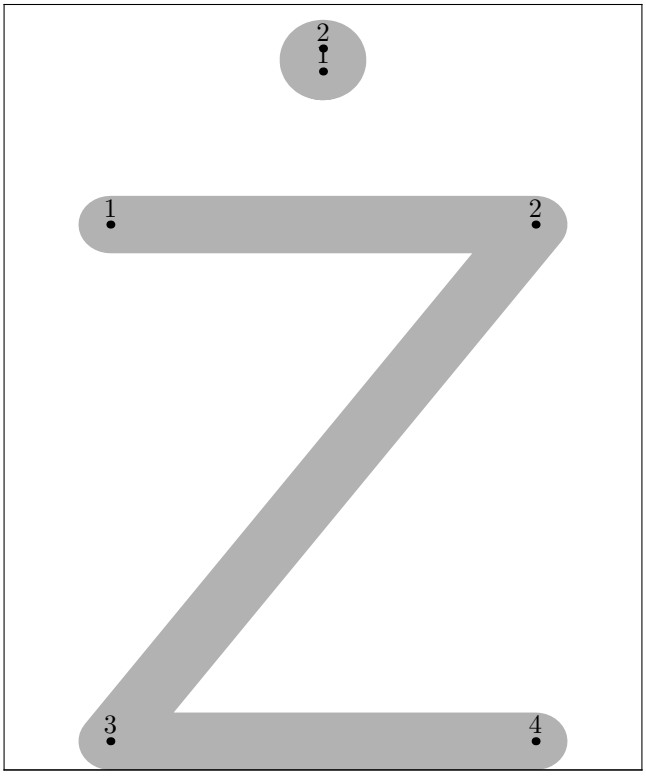
Glyph with coding number 153



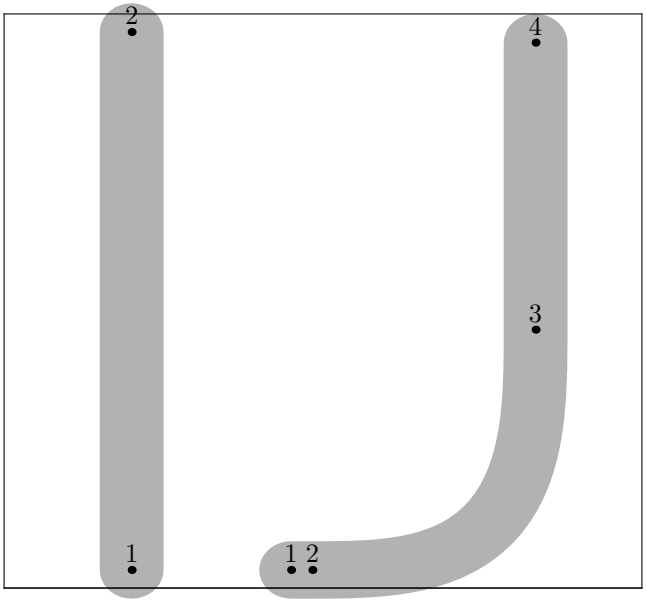
Glyph with coding number 154



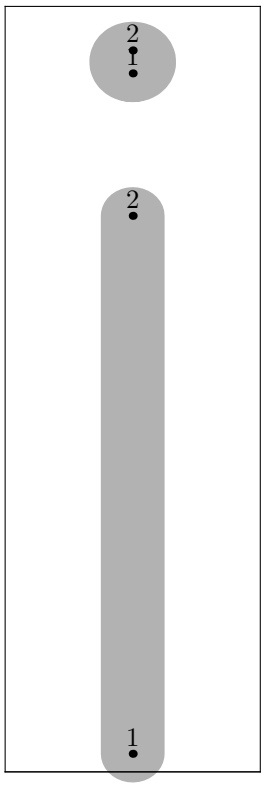
Glyph with coding number 155



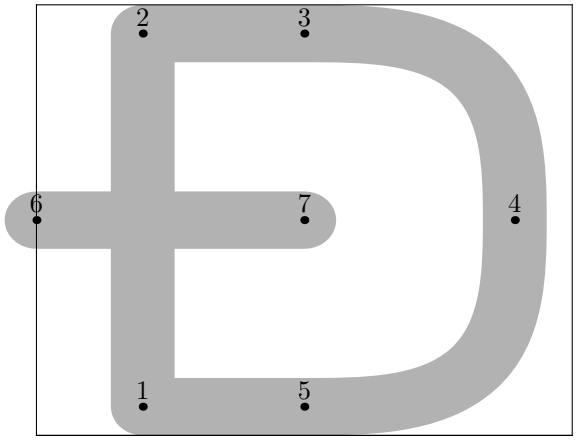
Glyph with coding number 156



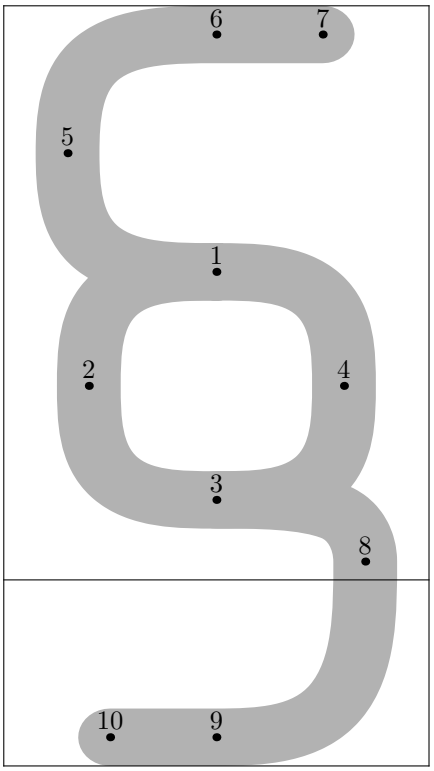
Glyph with coding number 157



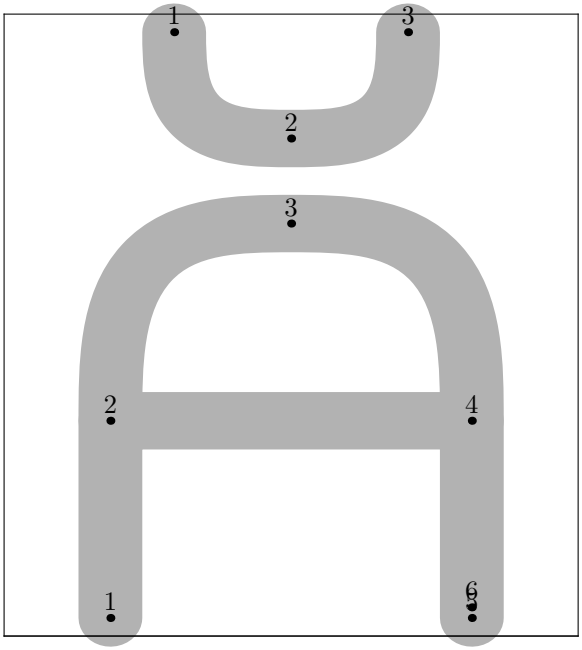
Glyph with coding number 158



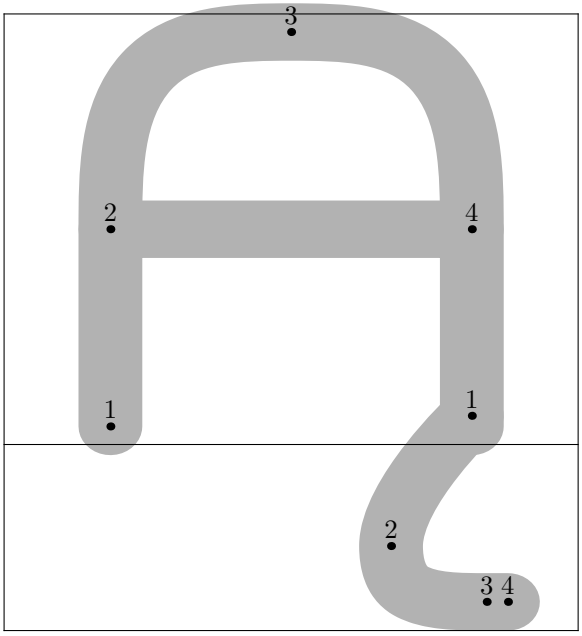
Glyph with coding number 159



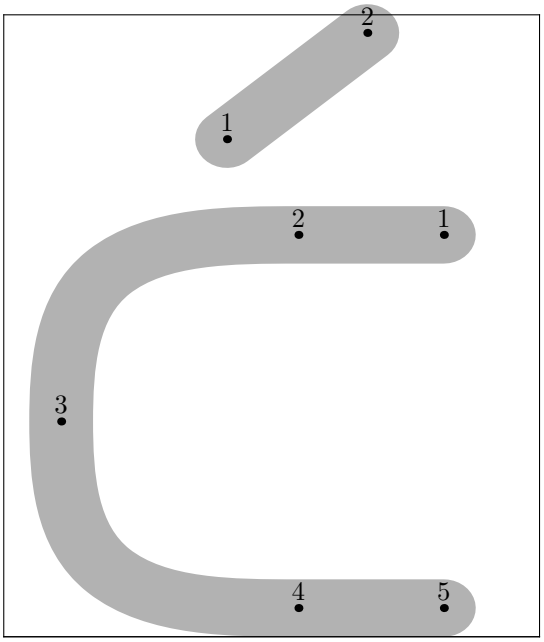
Glyph with coding number 160



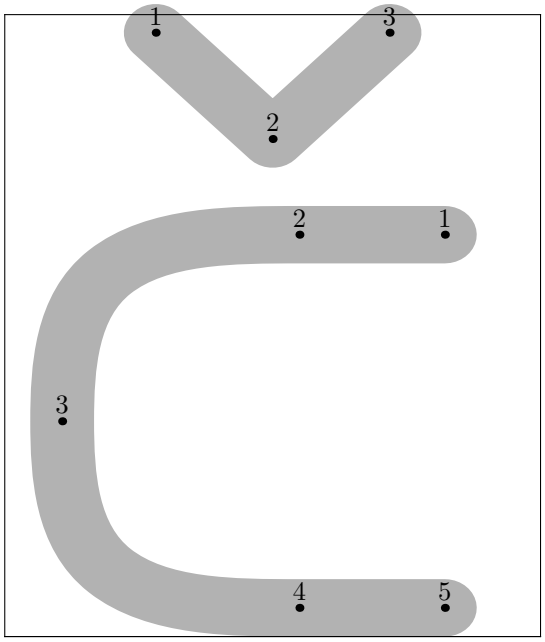
Glyph with coding number 161



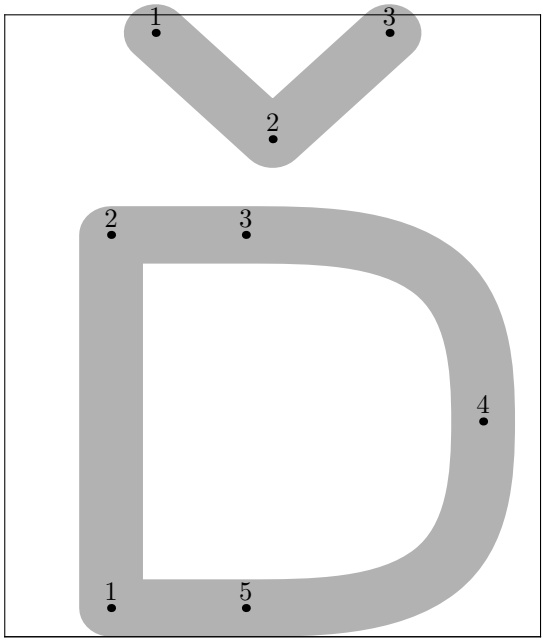
Glyph with coding number 162



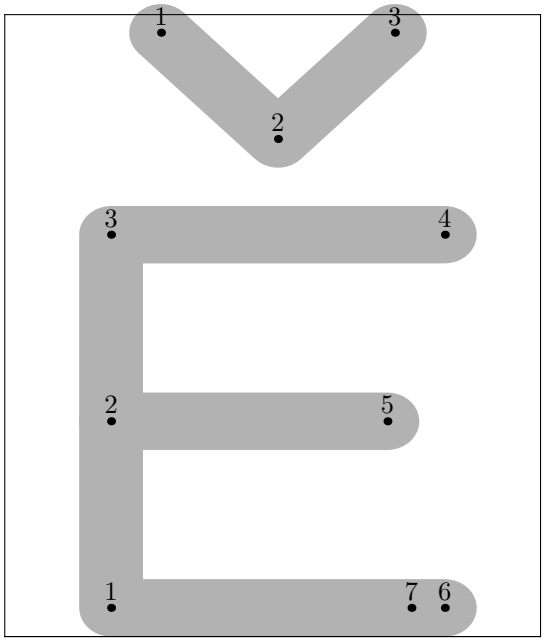
Glyph with coding number 163



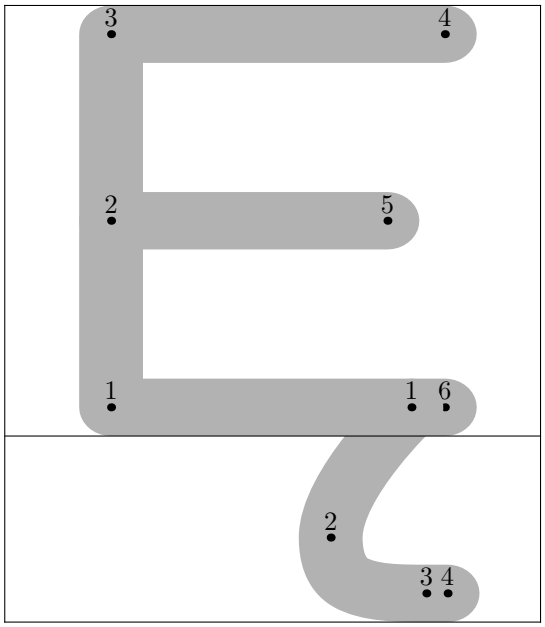
Glyph with coding number 164



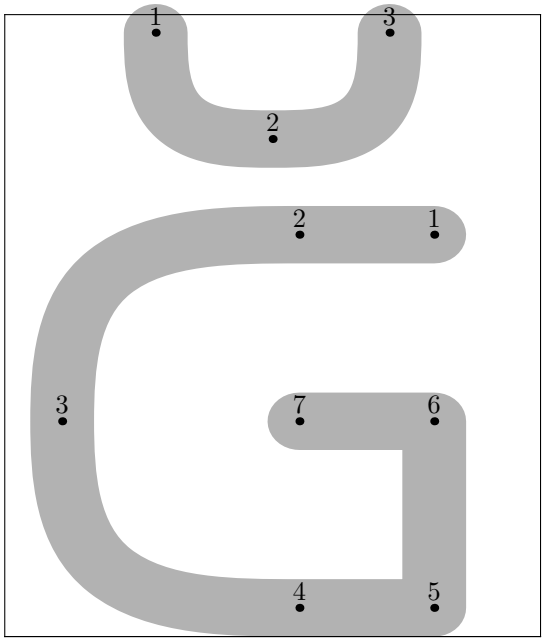
Glyph with coding number 165



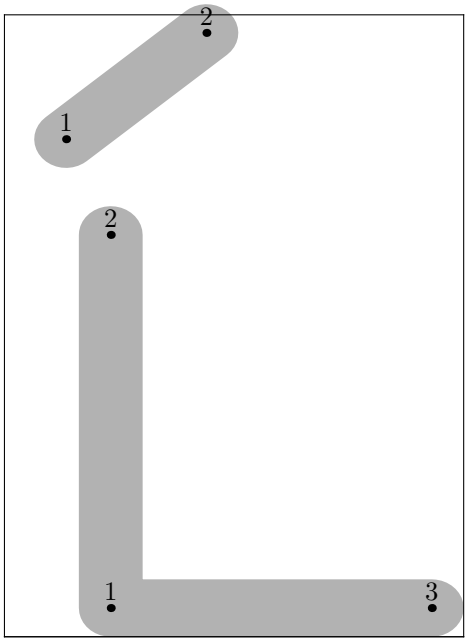
Glyph with coding number 166



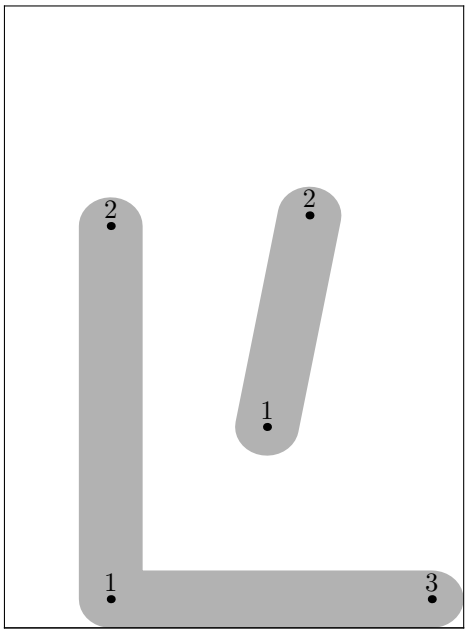
Glyph with coding number 167



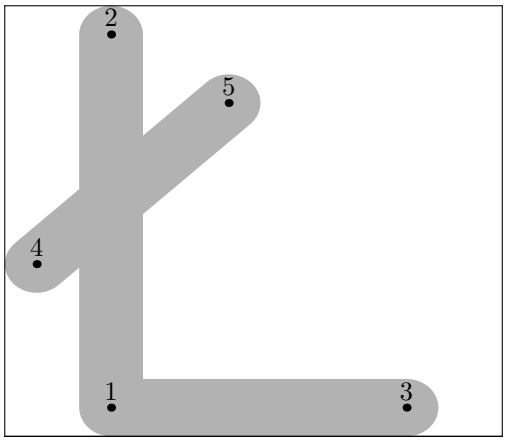
Glyph with coding number 168



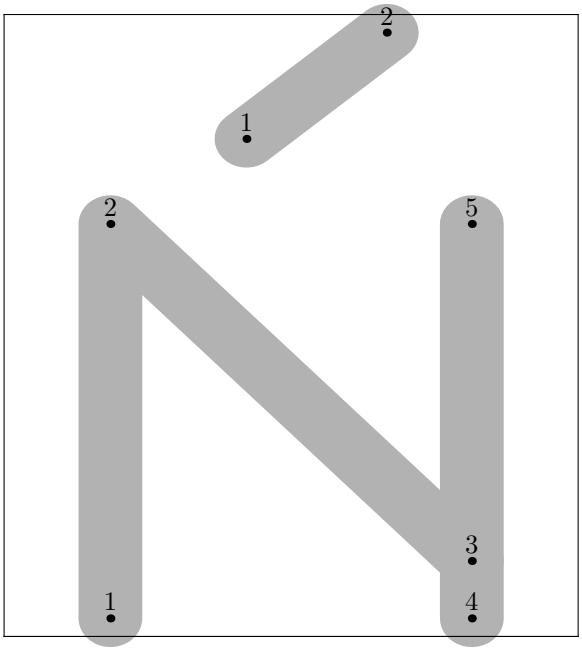
Glyph with coding number 169



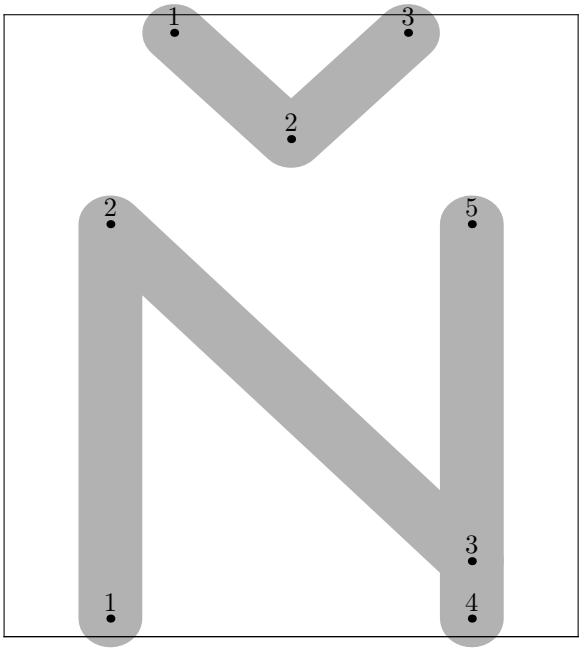
Glyph with coding number 170



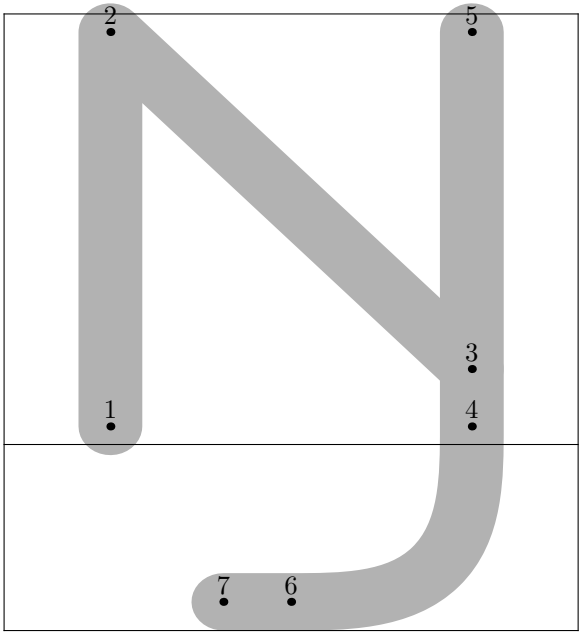
Glyph with coding number 171



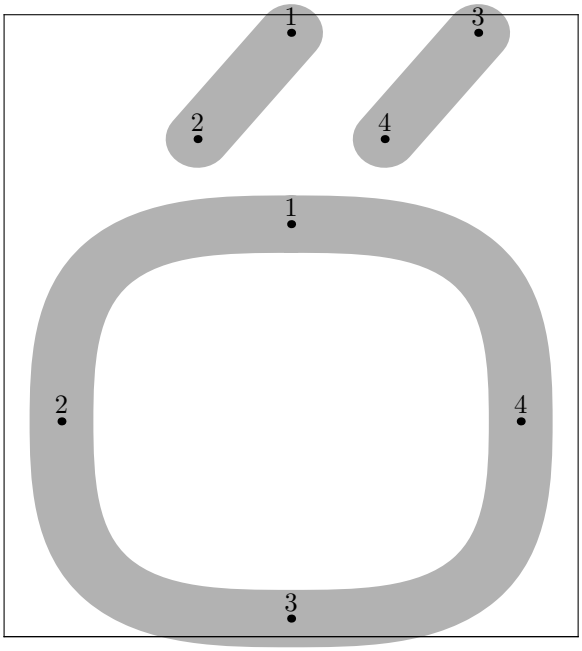
Glyph with coding number 172



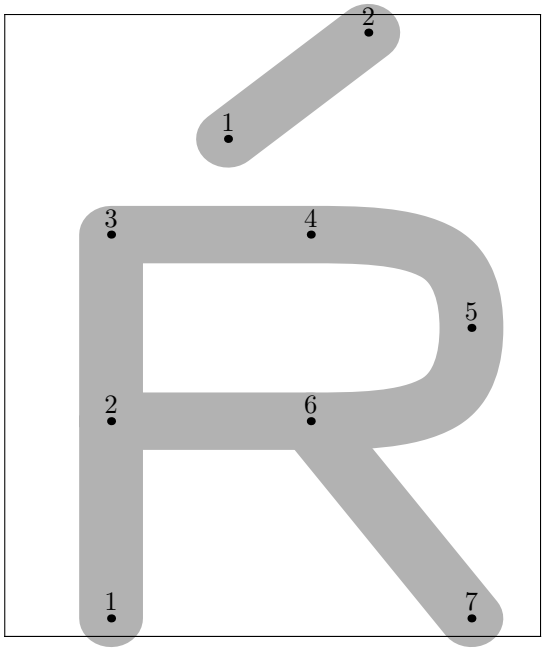
Glyph with coding number 173



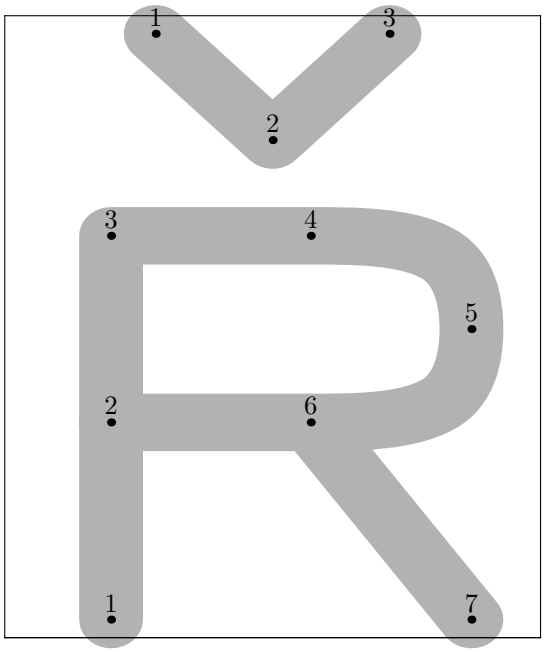
Glyph with coding number 174



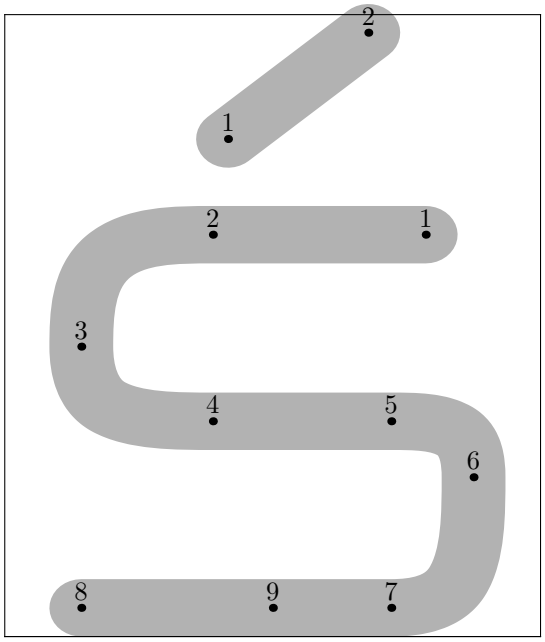
Glyph with coding number 175



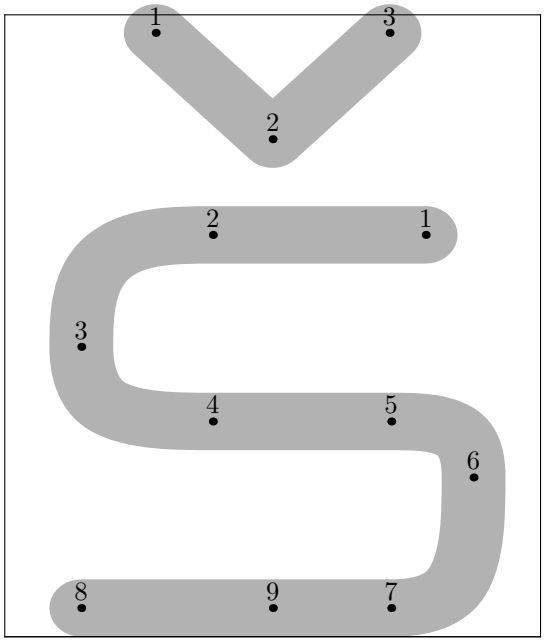
Glyph with coding number 176



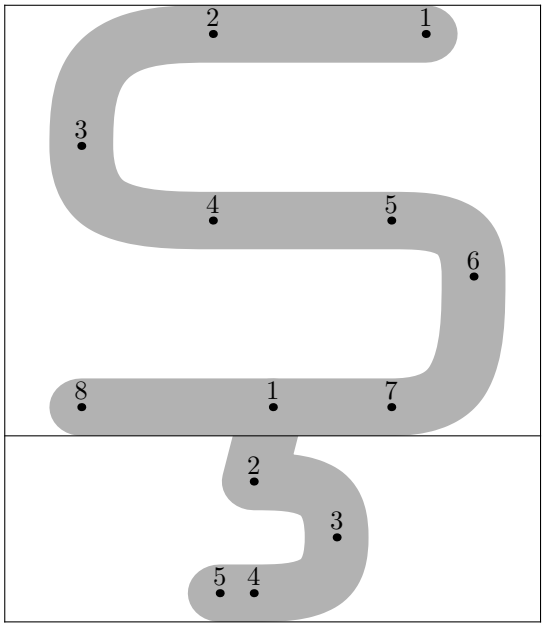
Glyph with coding number 177



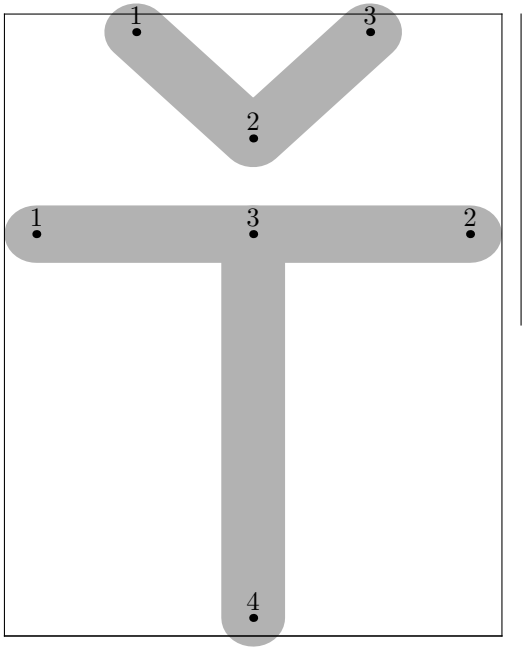
Glyph with coding number 178



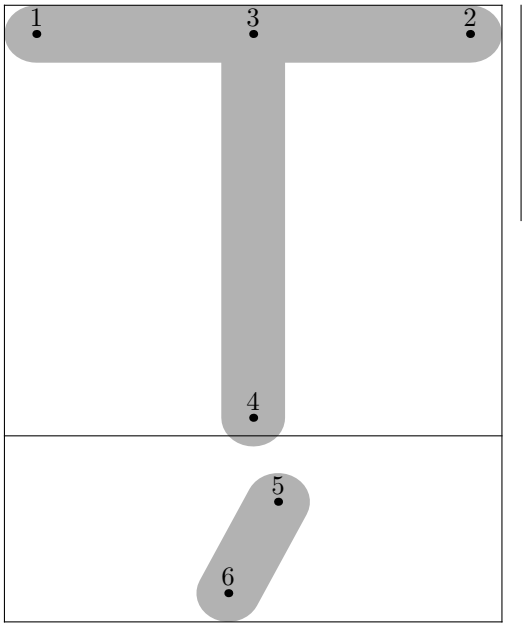
Glyph with coding number 179



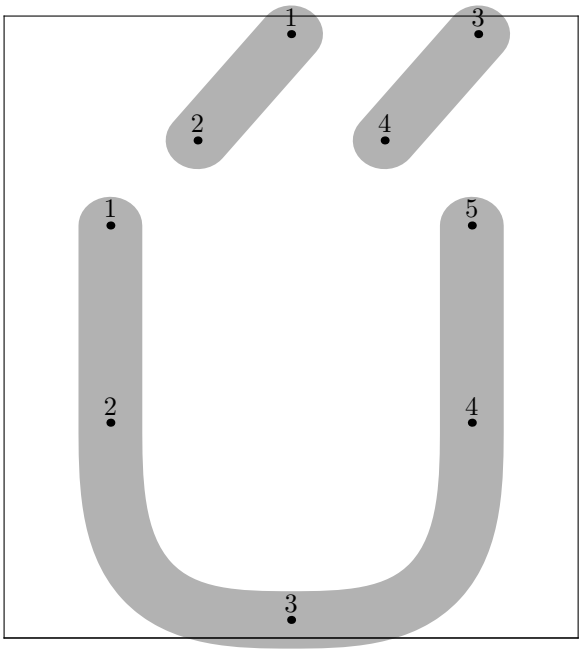
Glyph with coding number 180



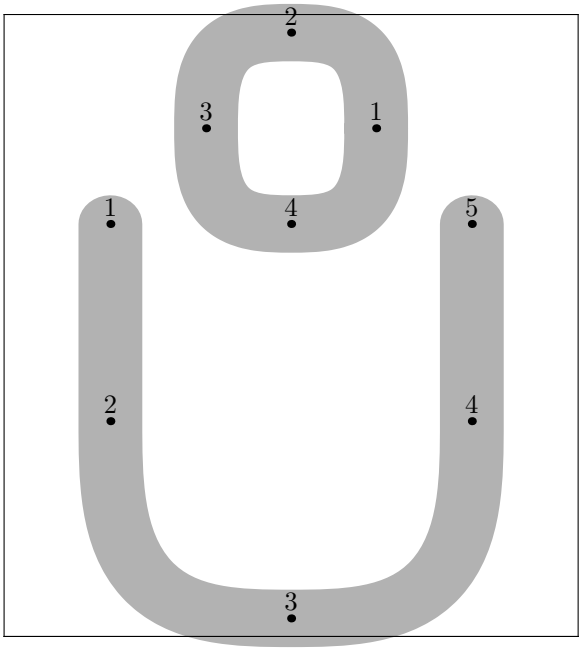
Glyph with coding number 181



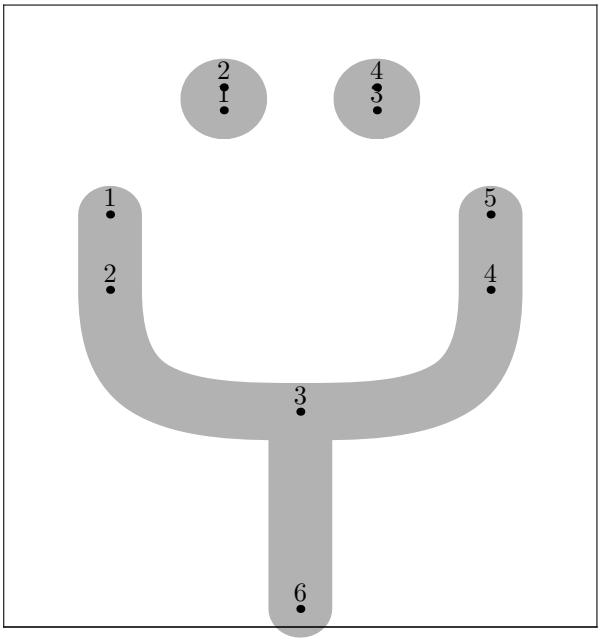
Glyph with coding number 182



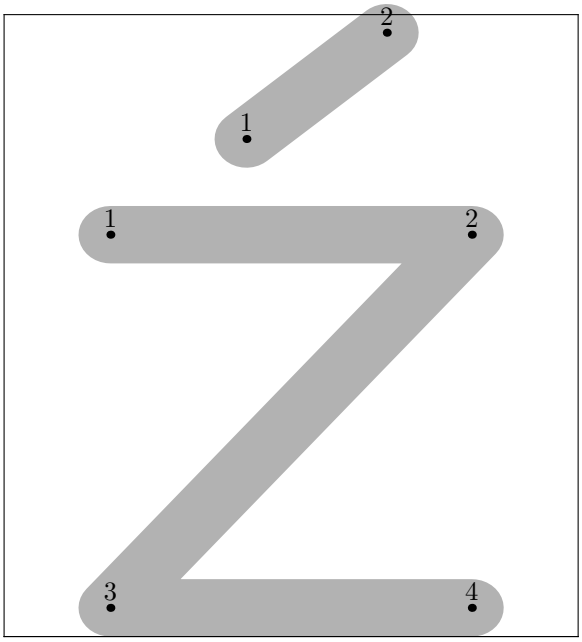
Glyph with coding number 183



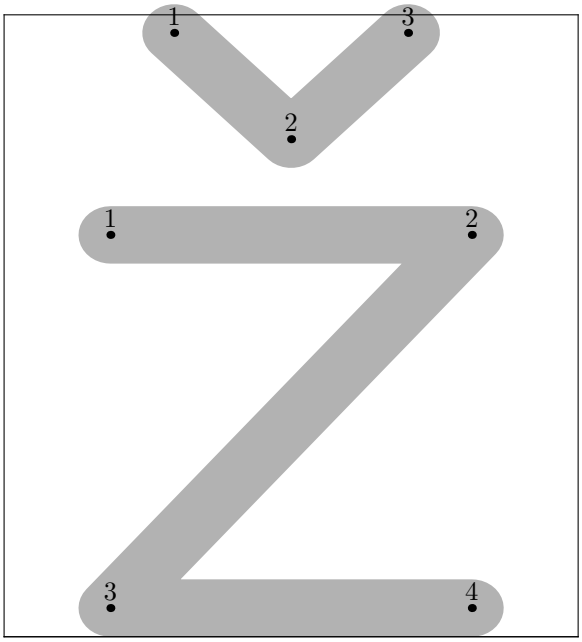
Glyph with coding number 184



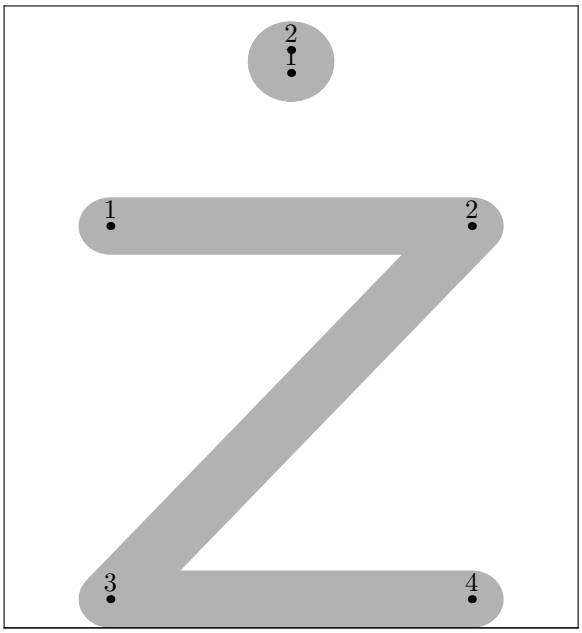
Glyph with coding number 185



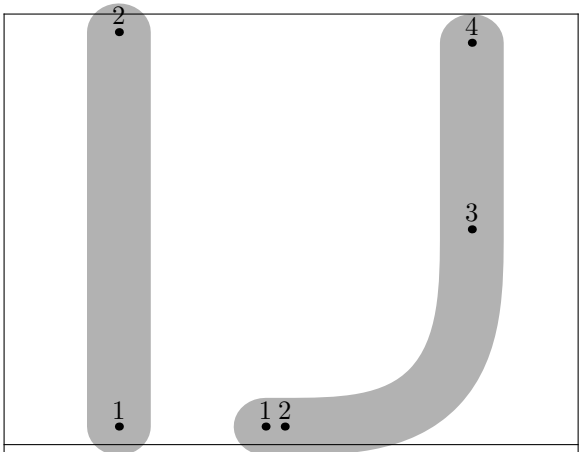
Glyph with coding number 186



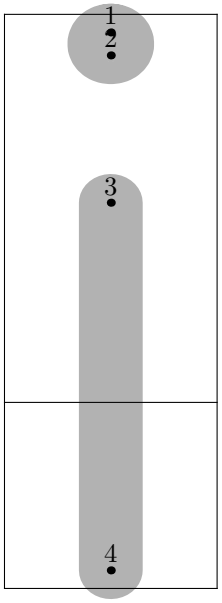
Glyph with coding number 187



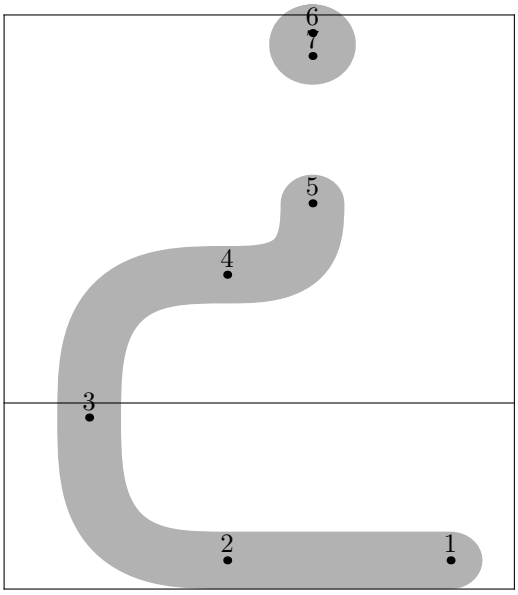
Glyph with coding number 188



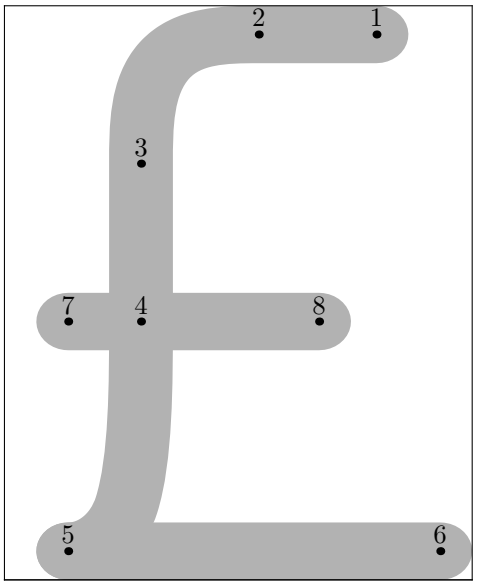
Glyph with coding number 189



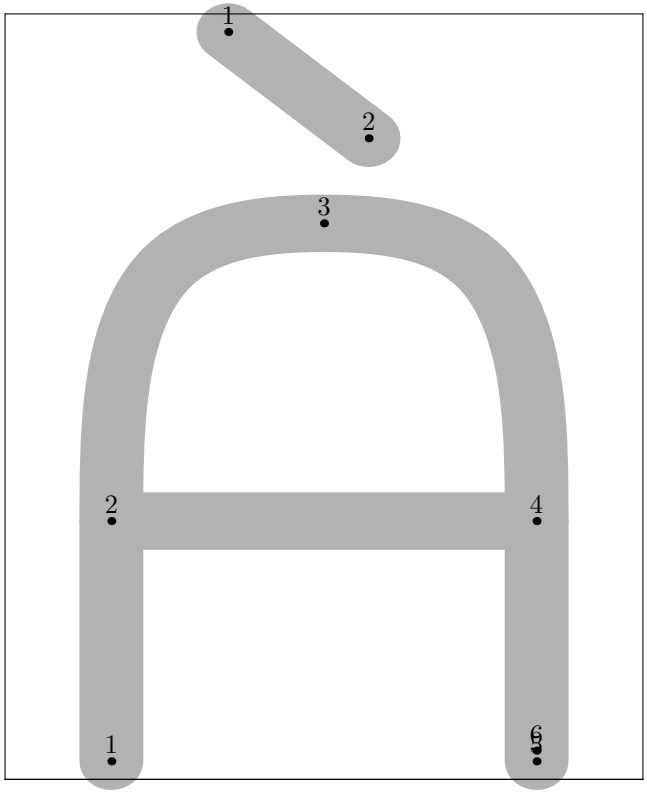
Glyph with coding number 190



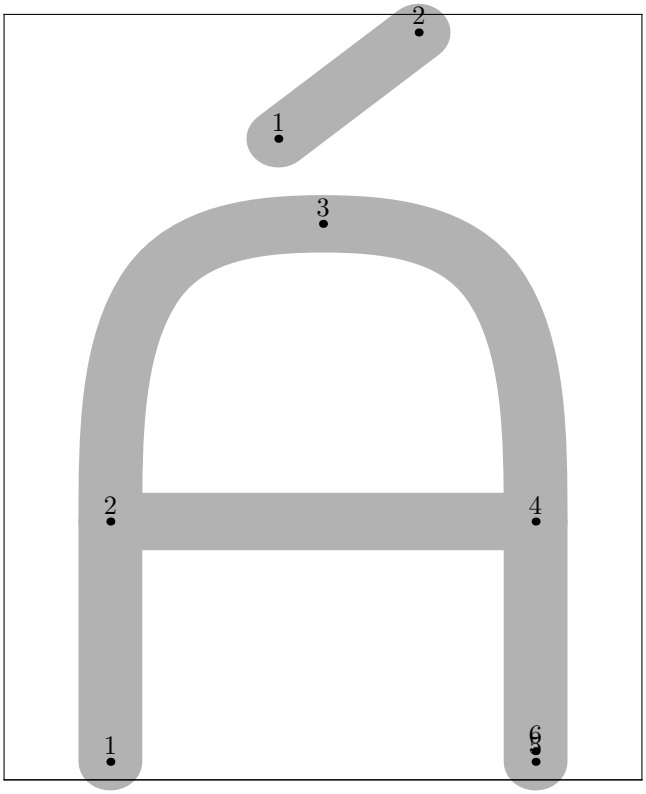
Glyph with coding number 191



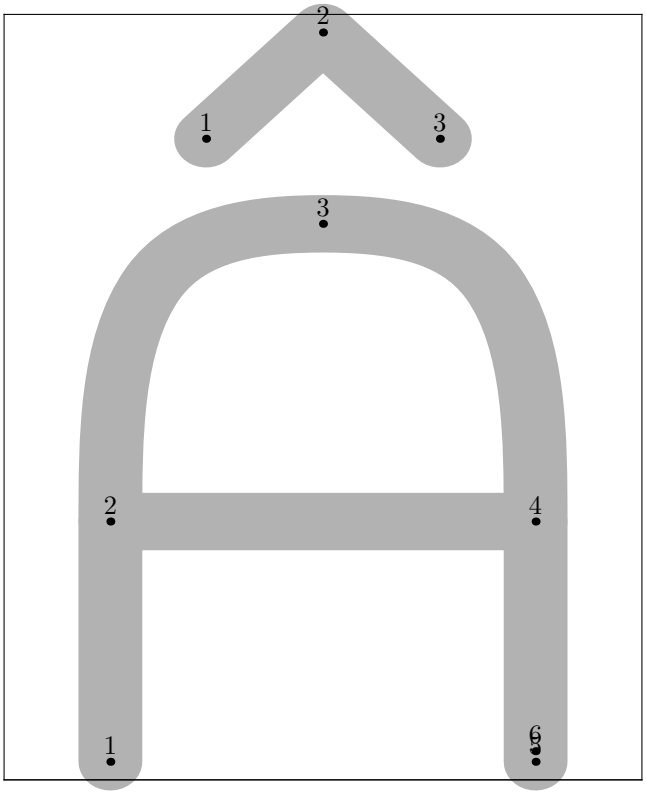
Glyph with coding number 192



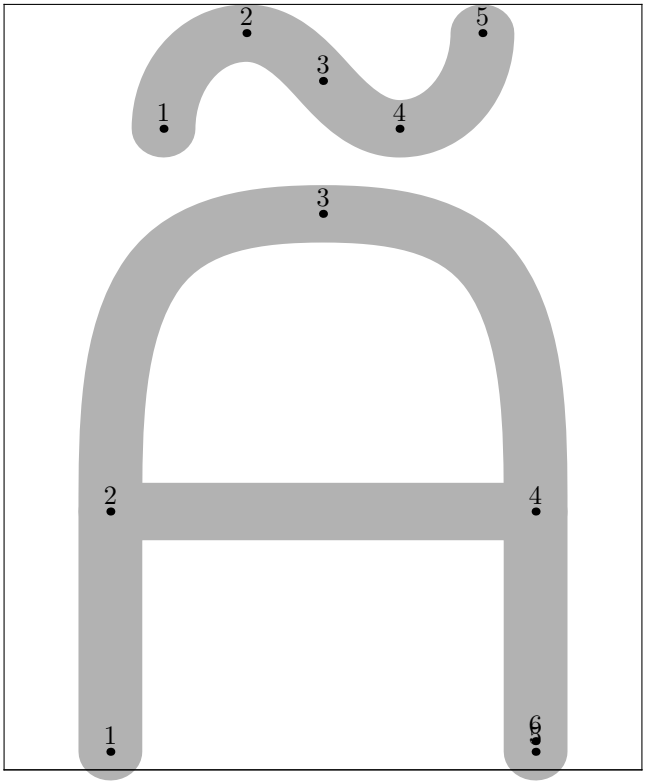
Glyph with coding number 193



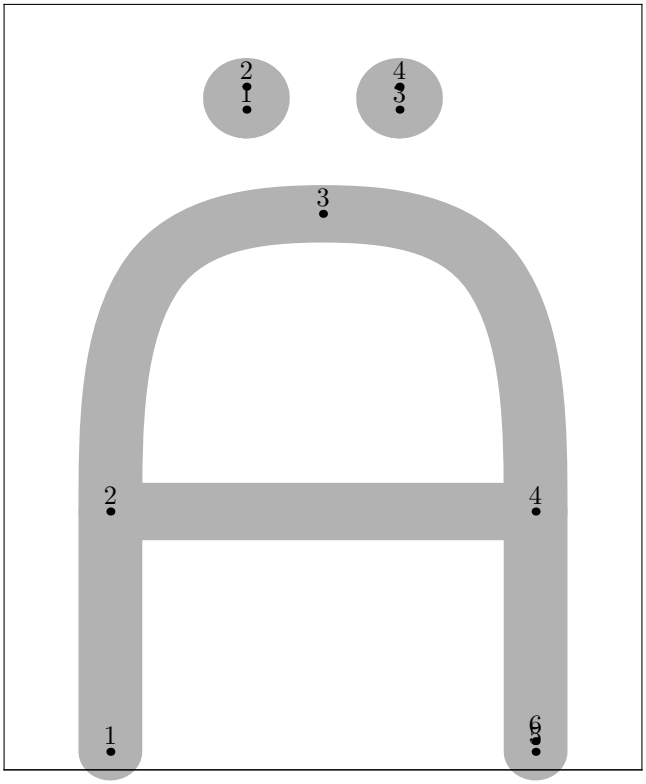
Glyph with coding number 194



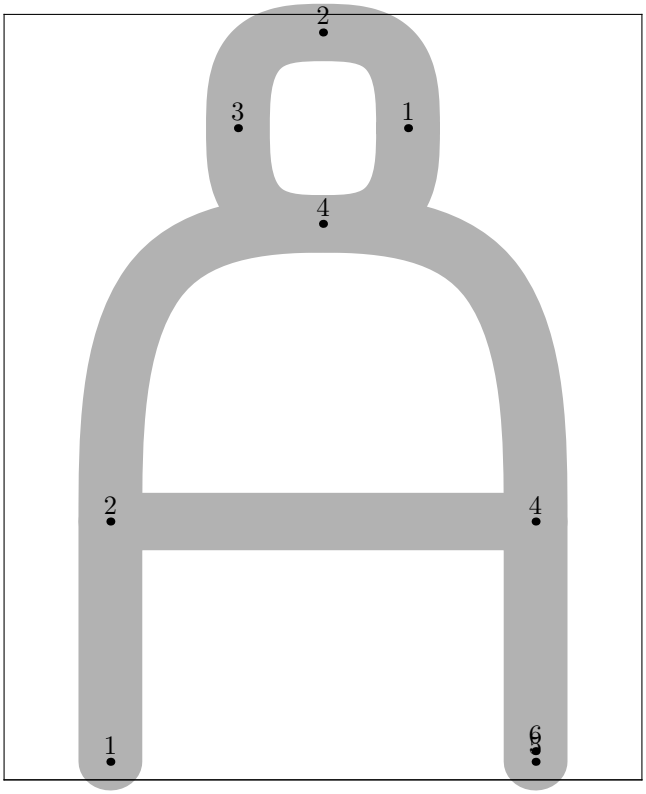
Glyph with coding number 195



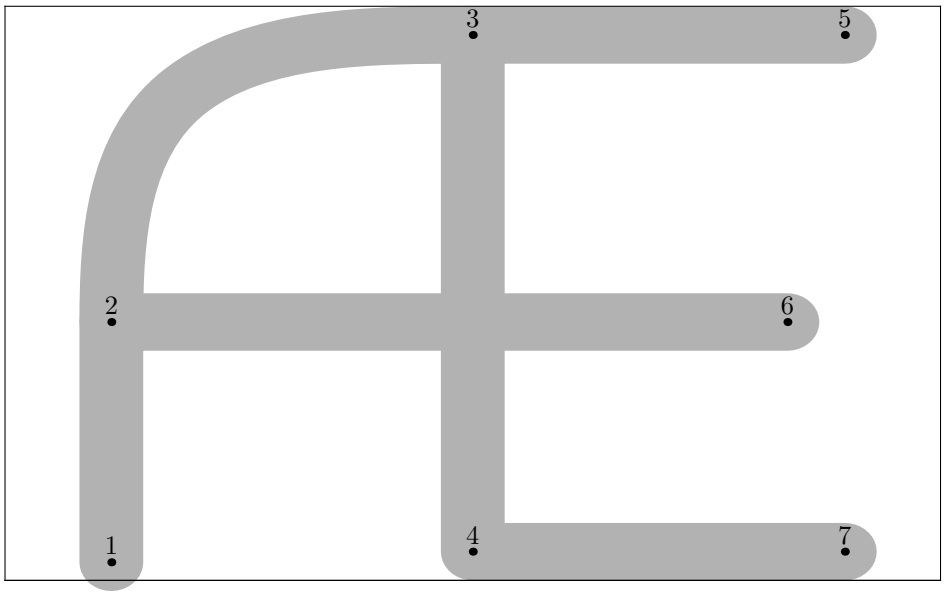
Glyph with coding number 196



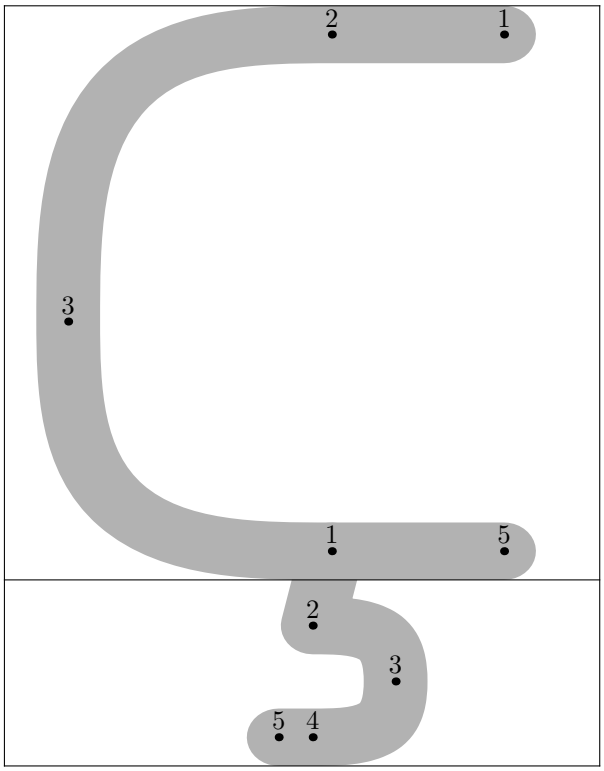
Glyph with coding number 197



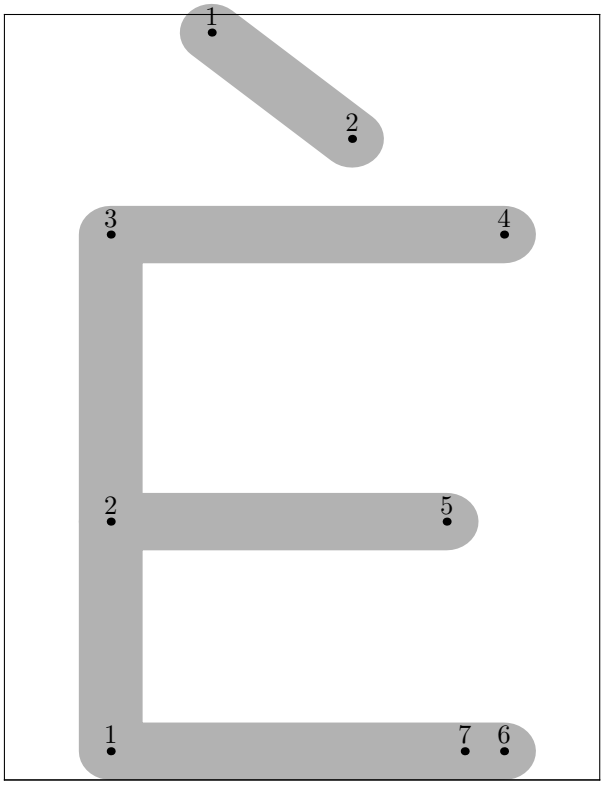
Glyph with coding number 198



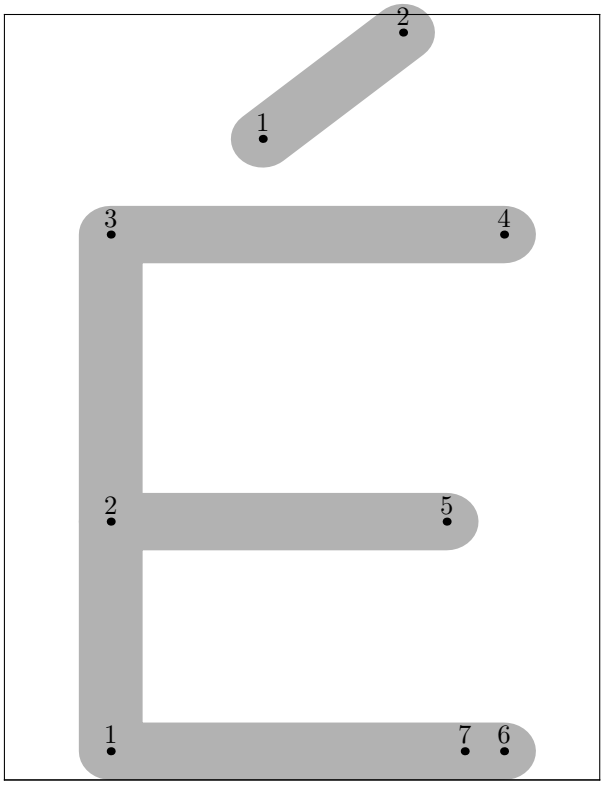
Glyph with coding number 199



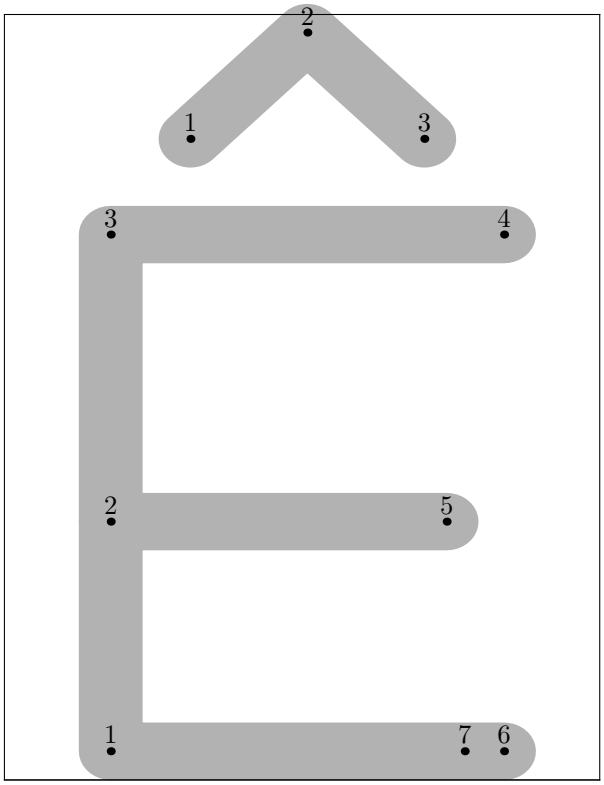
Glyph with coding number 200



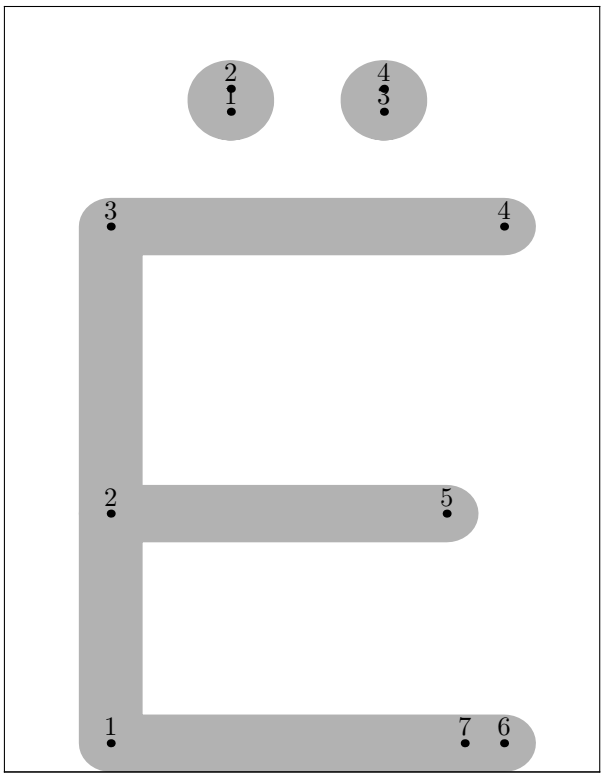
Glyph with coding number 201



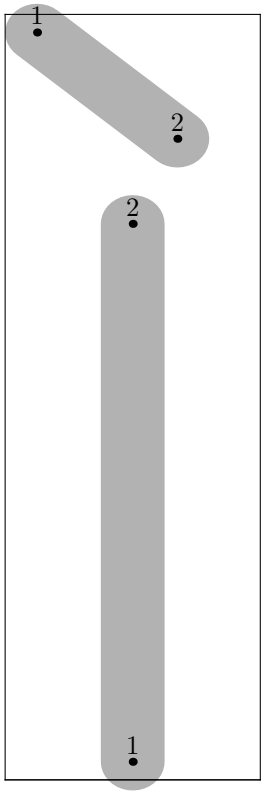
Glyph with coding number 202



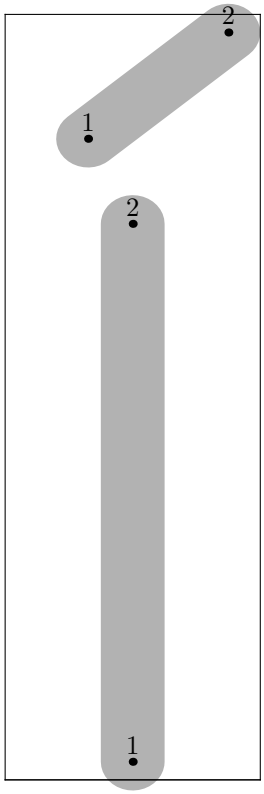
Glyph with coding number 203



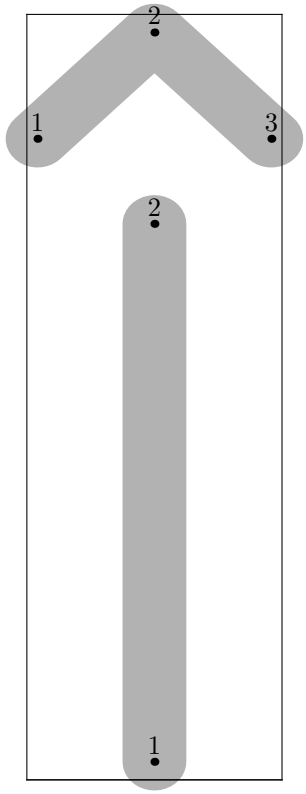
Glyph with coding number 204



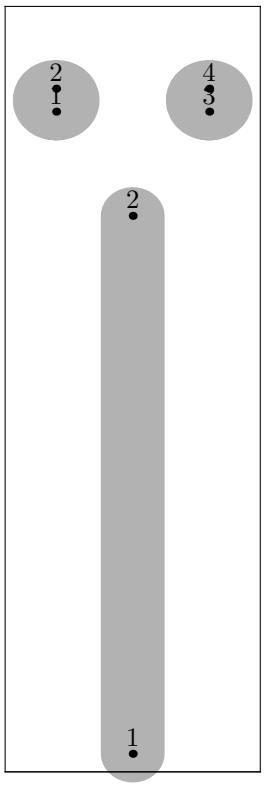
Glyph with coding number 205



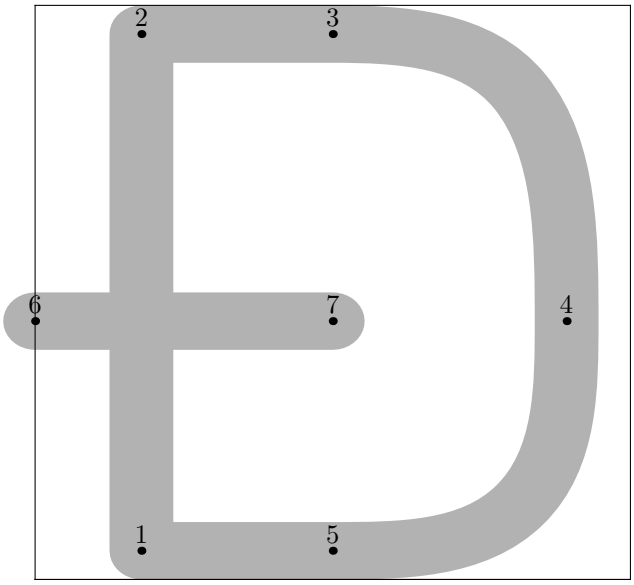
Glyph with coding number 206



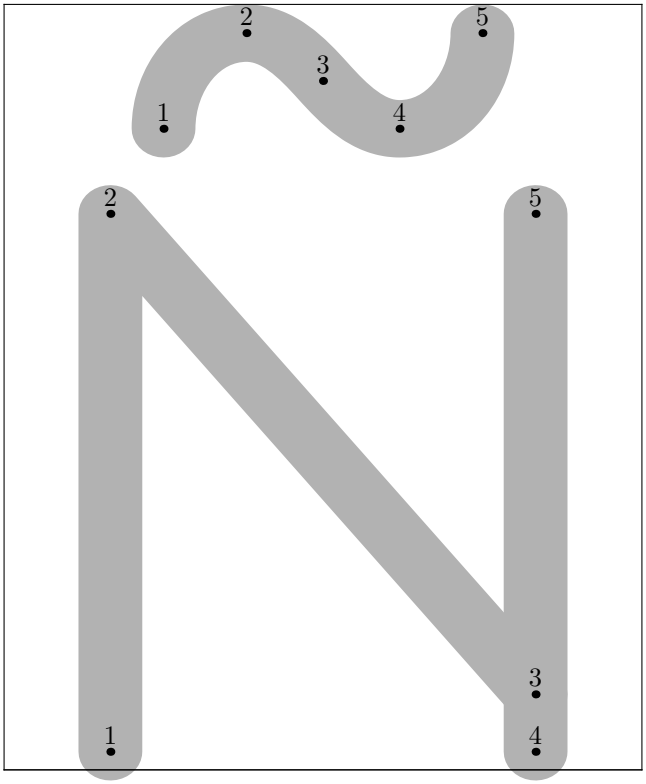
Glyph with coding number 207



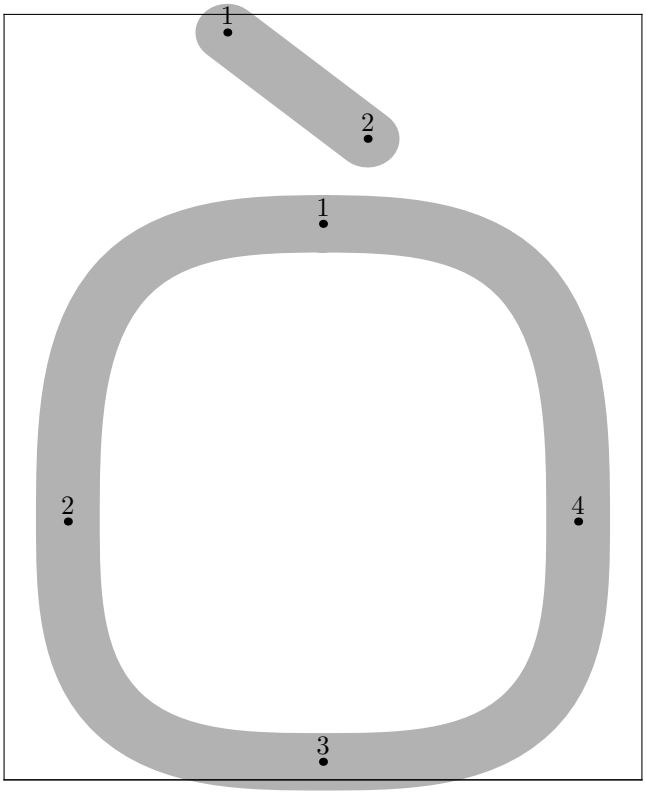
Glyph with coding number 208



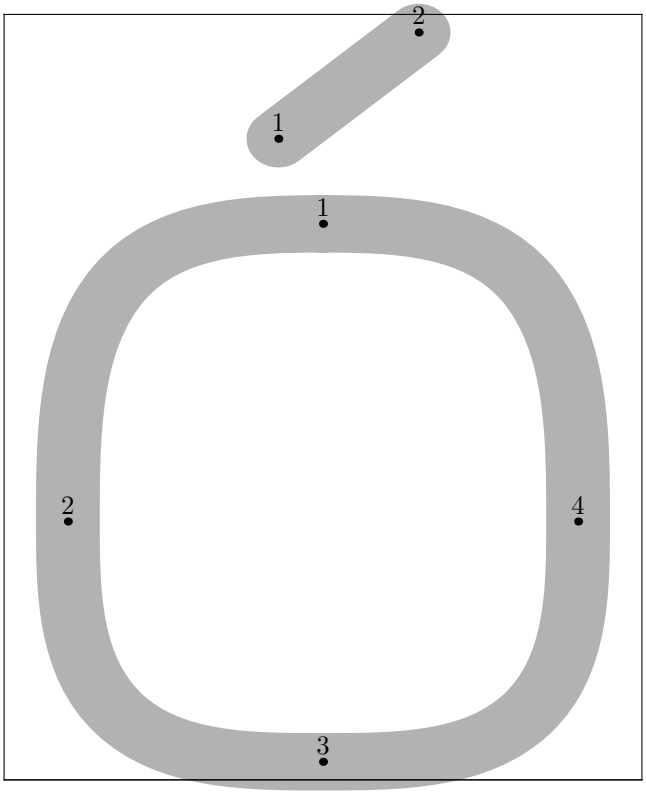
Glyph with coding number 209



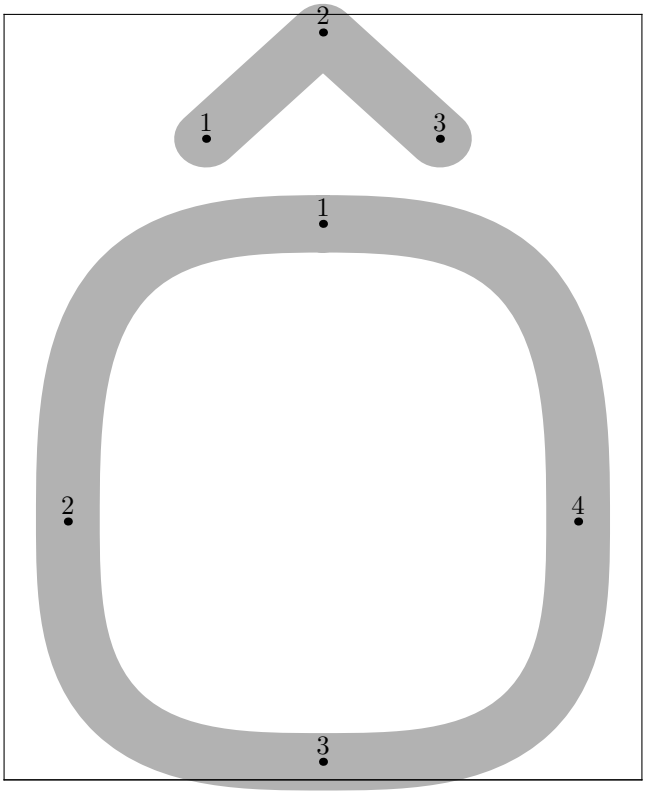
Glyph with coding number 210



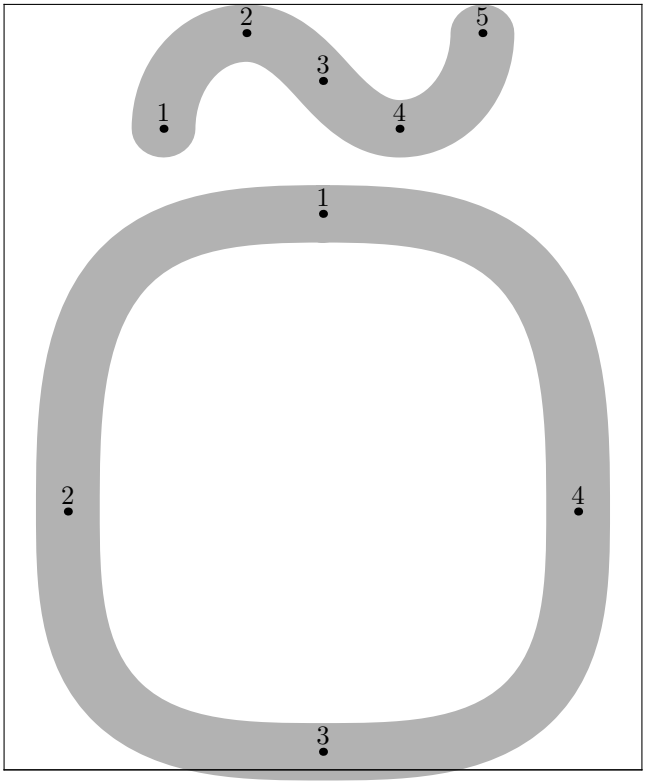
Glyph with coding number 211



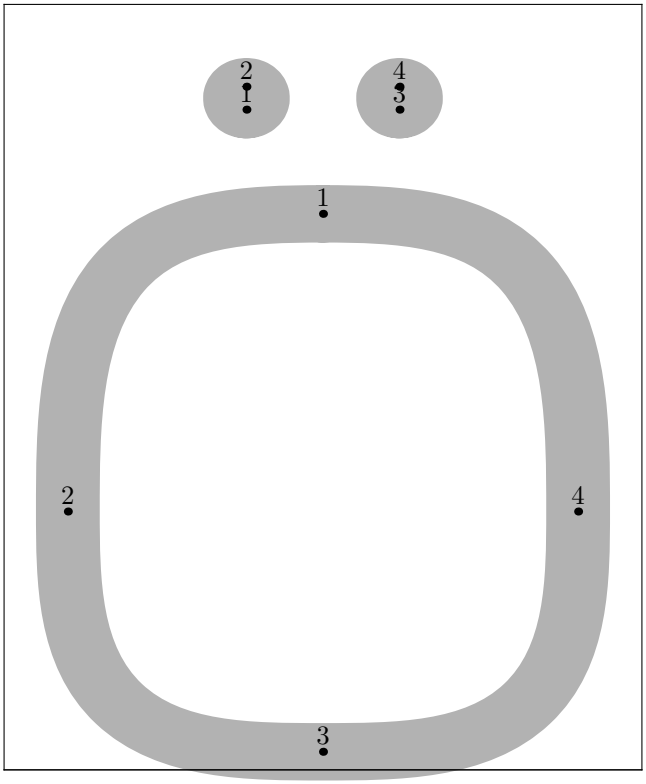
Glyph with coding number 212



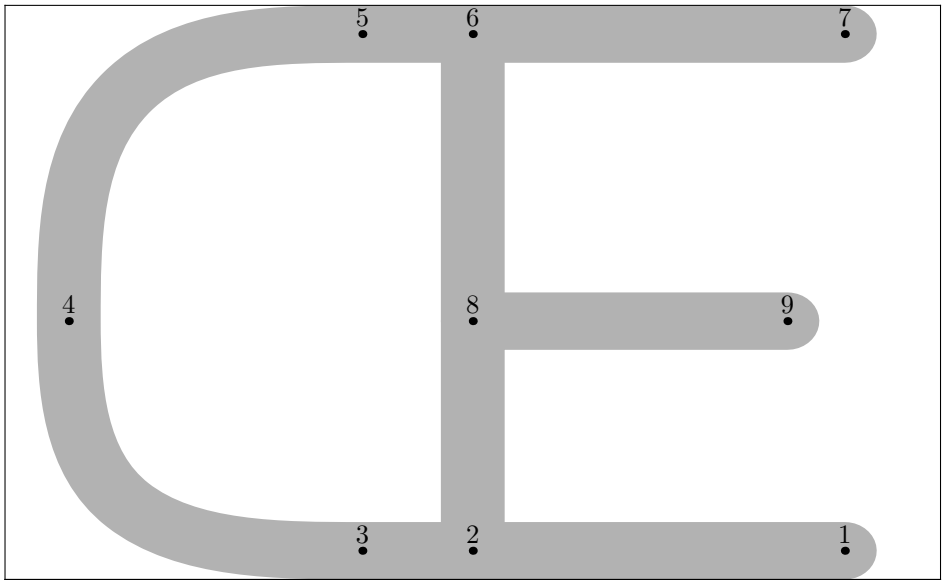
Glyph with coding number 213



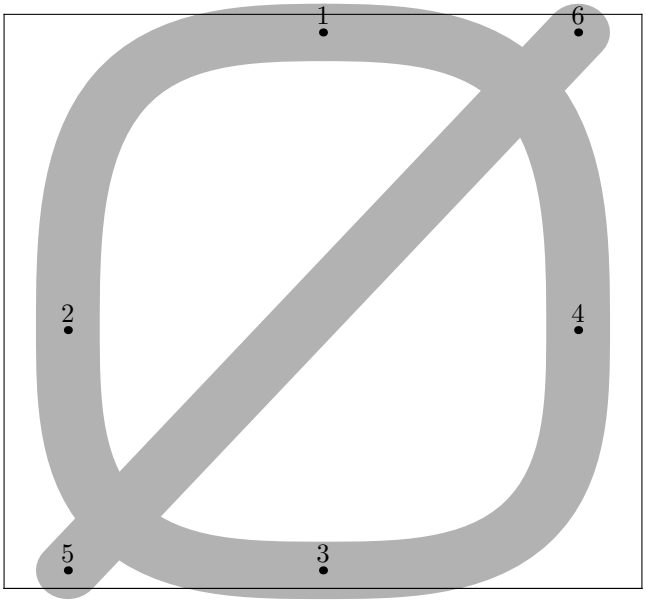
Glyph with coding number 214



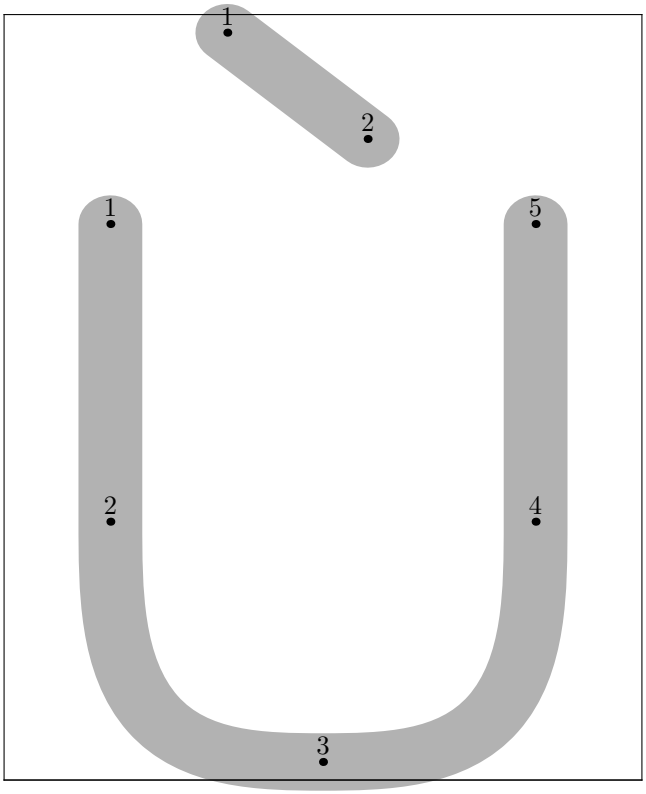
Glyph with coding number 215



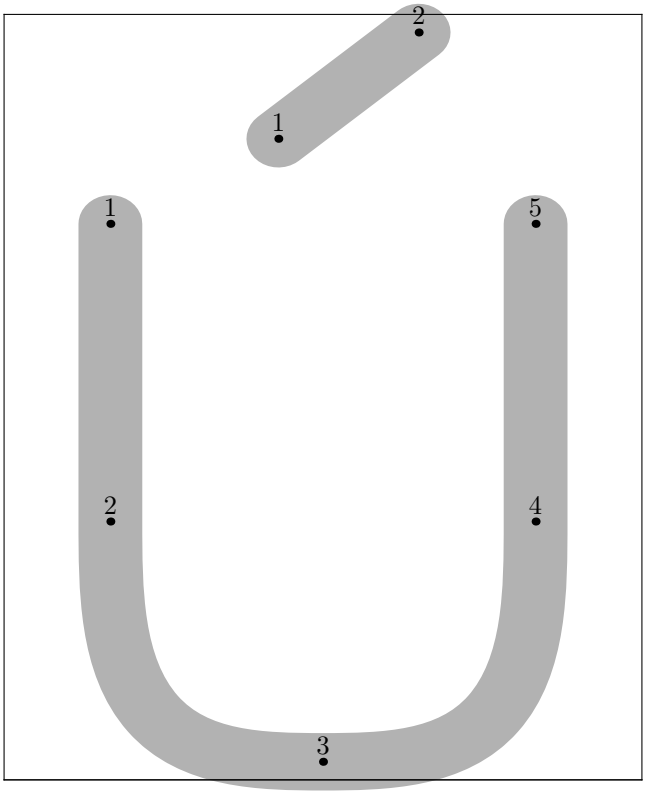
Glyph with coding number 216



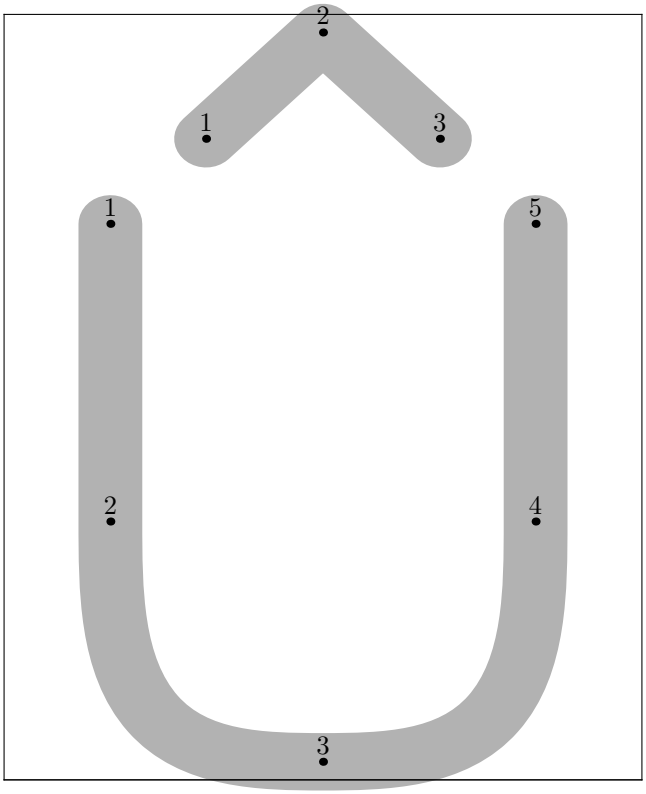
Glyph with coding number 217



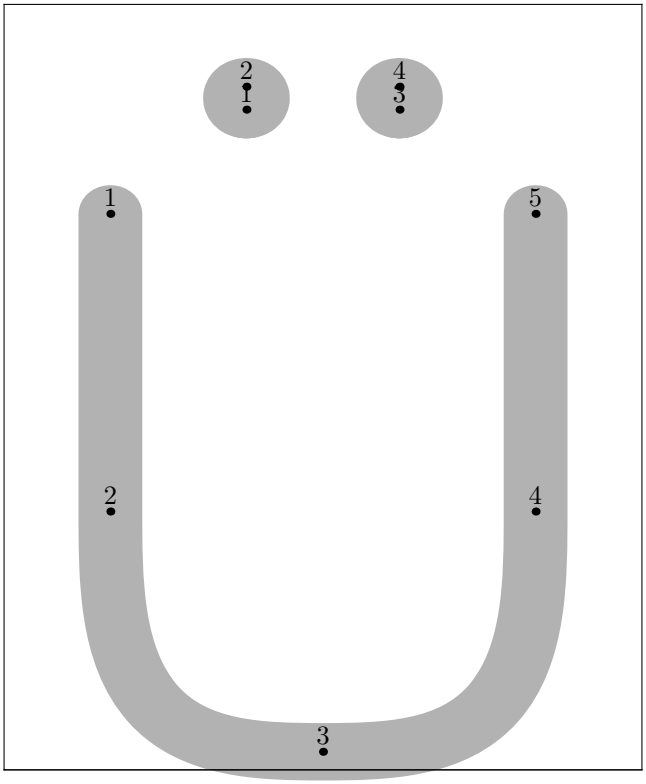
Glyph with coding number 218



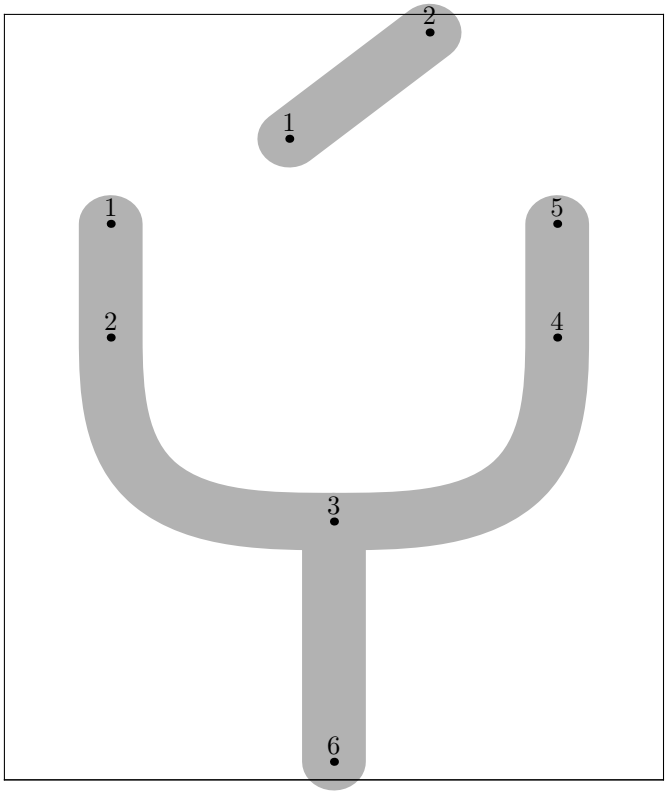
Glyph with coding number 219



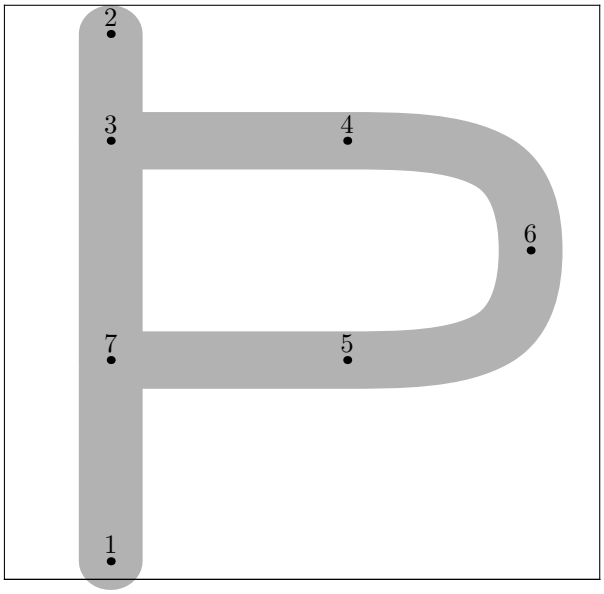
Glyph with coding number 220



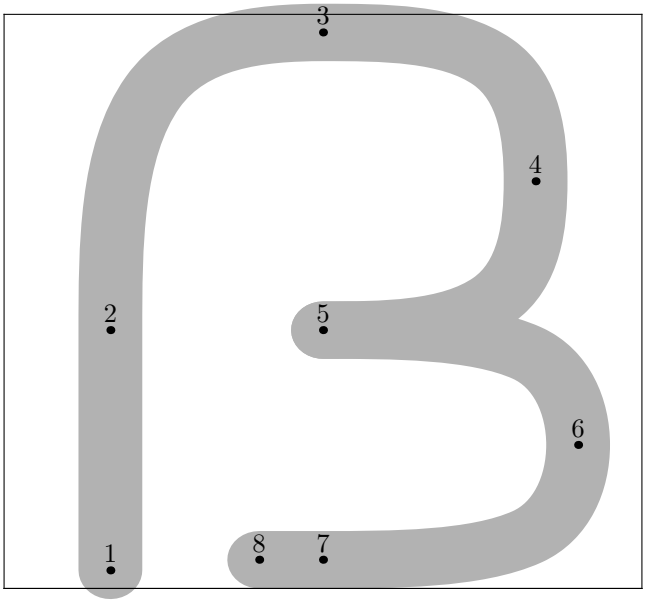
Glyph with coding number 221



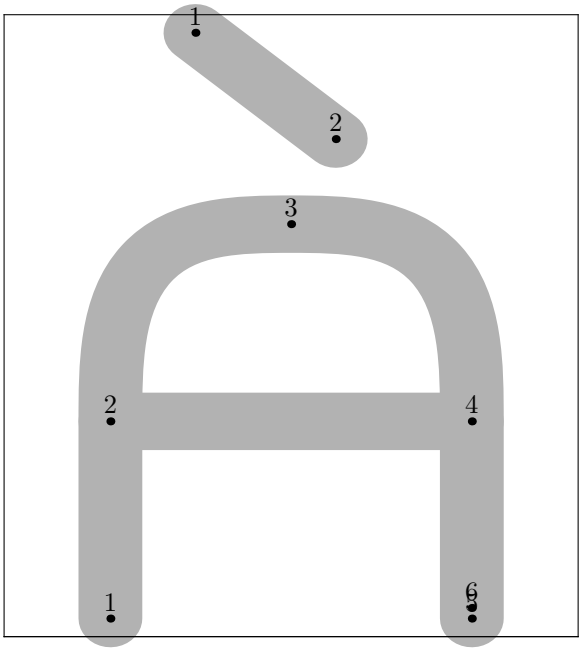
Glyph with coding number 222



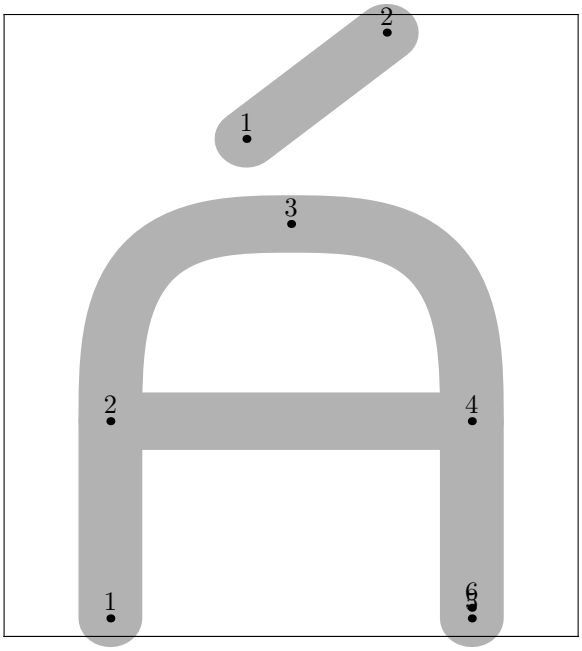
Glyph with coding number 223



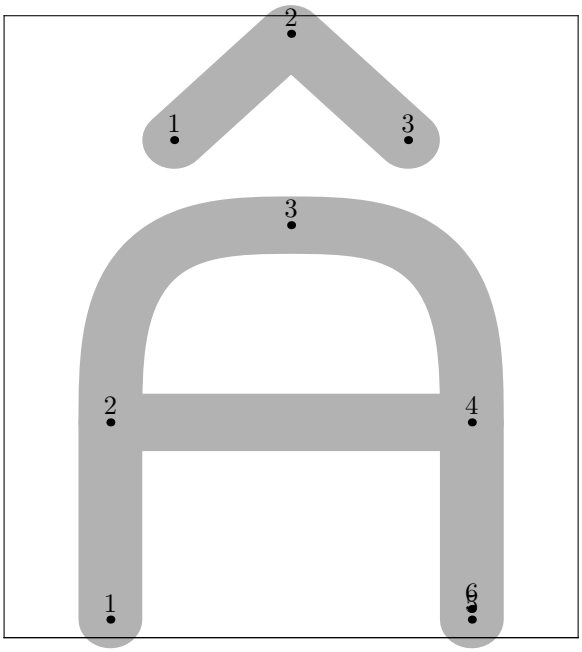
Glyph with coding number 224



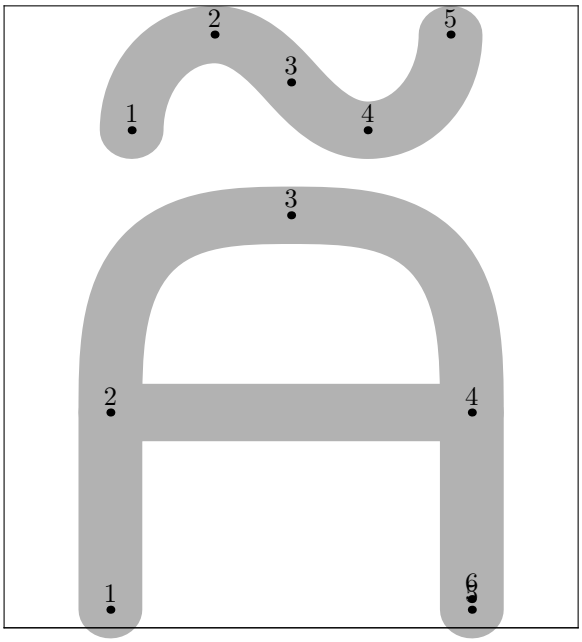
Glyph with coding number 225



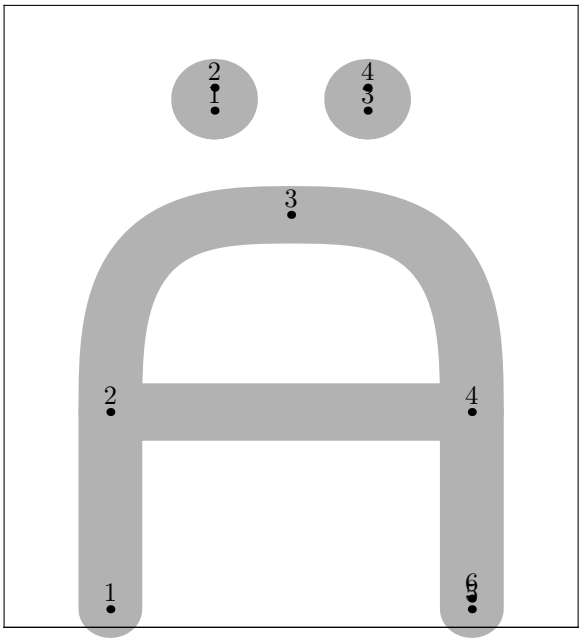
Glyph with coding number 226



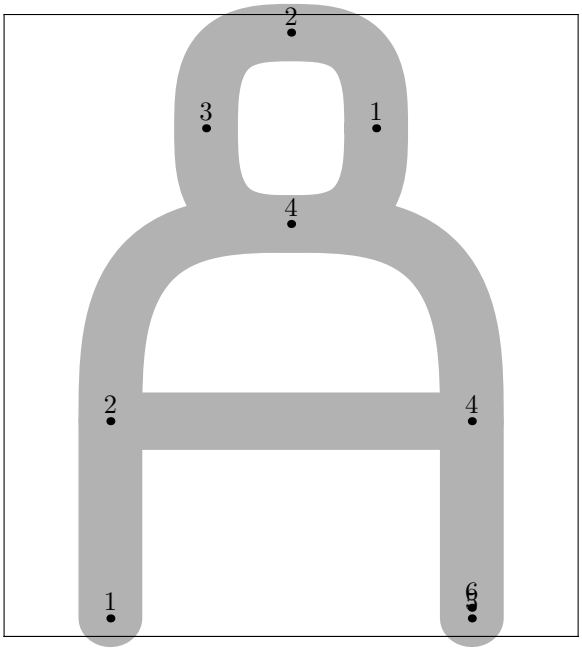
Glyph with coding number 227



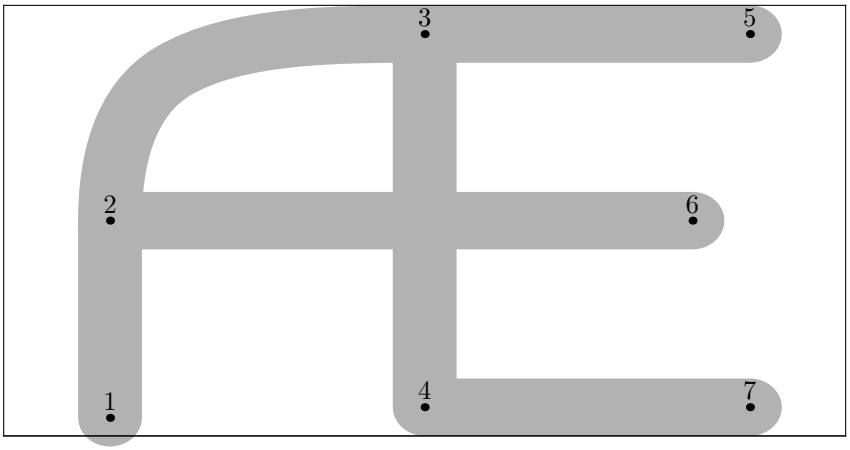
Glyph with coding number 228



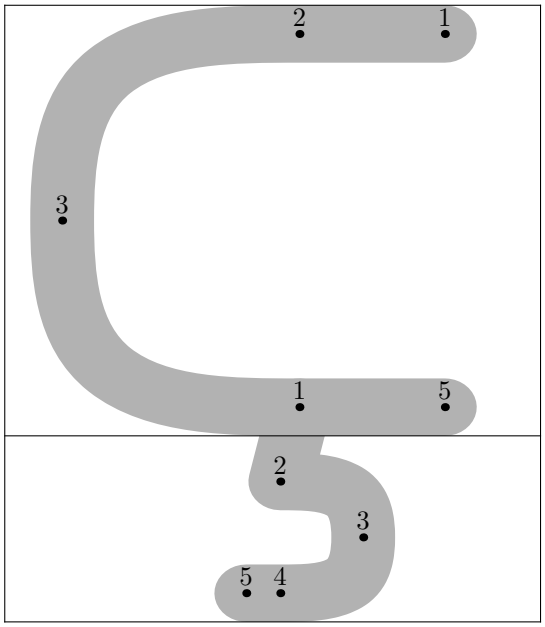
Glyph with coding number 229



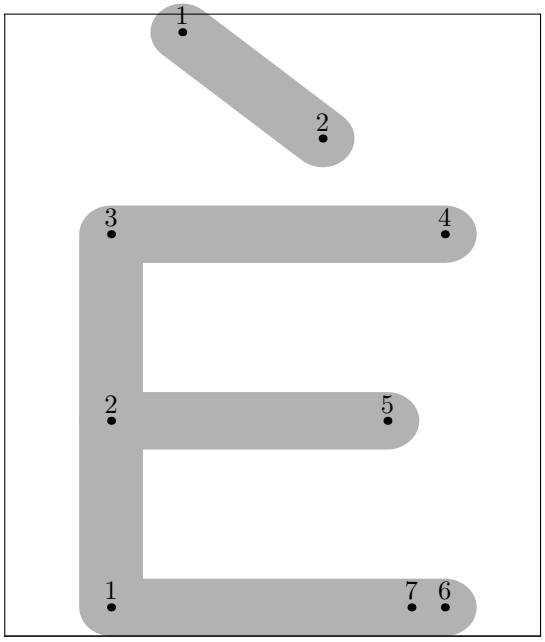
Glyph with coding number 230



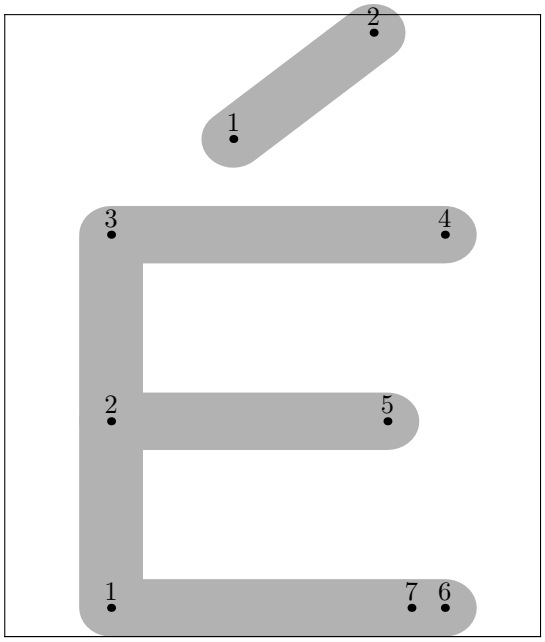
Glyph with coding number 231



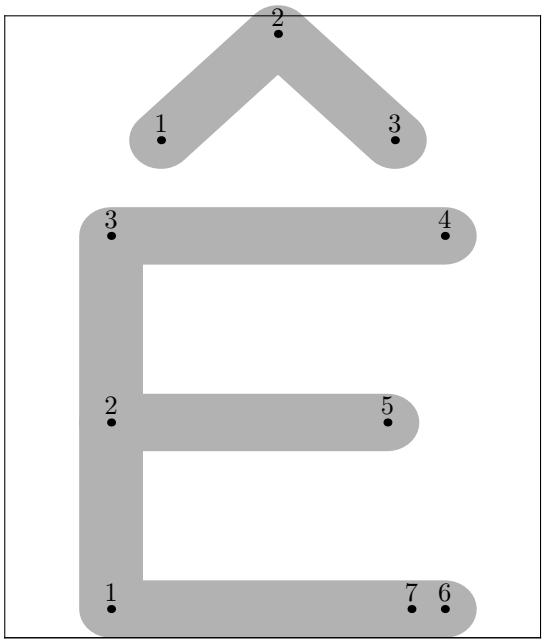
Glyph with coding number 232



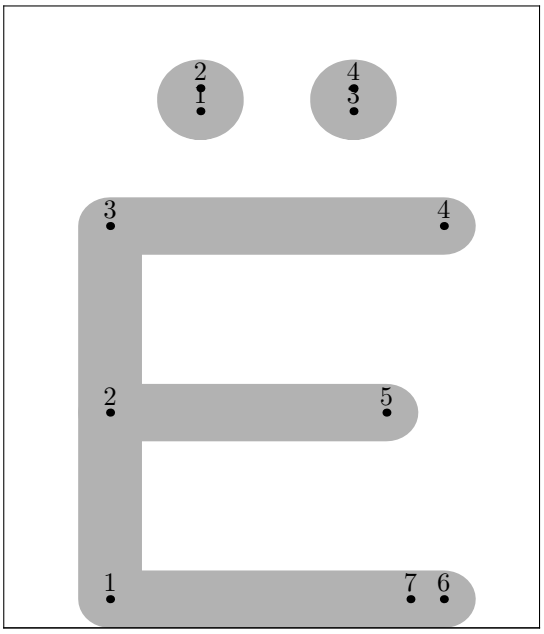
Glyph with coding number 233



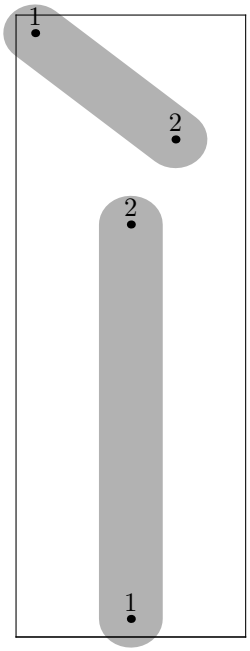
Glyph with coding number 234



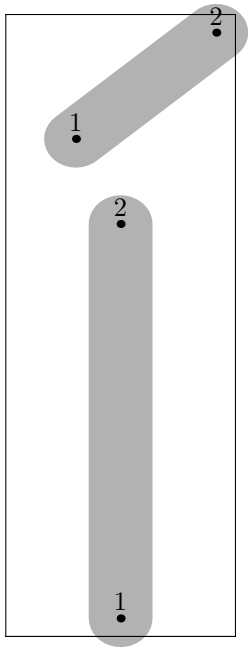
Glyph with coding number 235



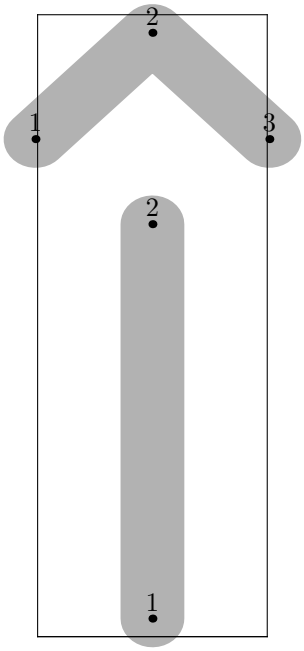
Glyph with coding number 236



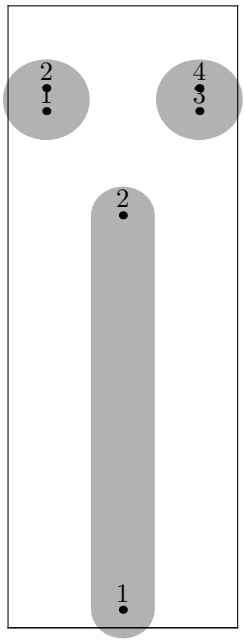
Glyph with coding number 237



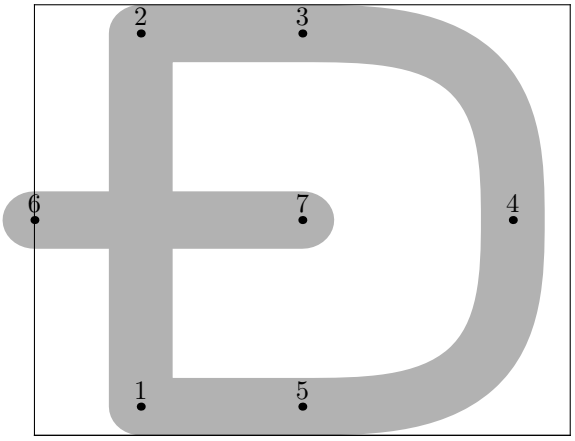
Glyph with coding number 238



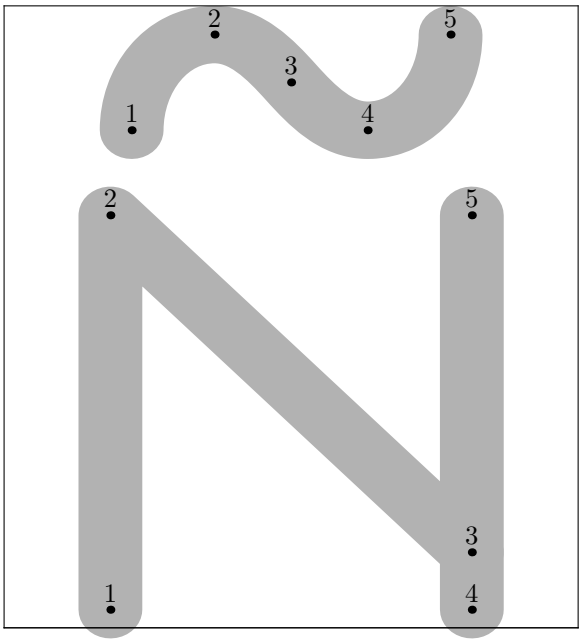
Glyph with coding number 239



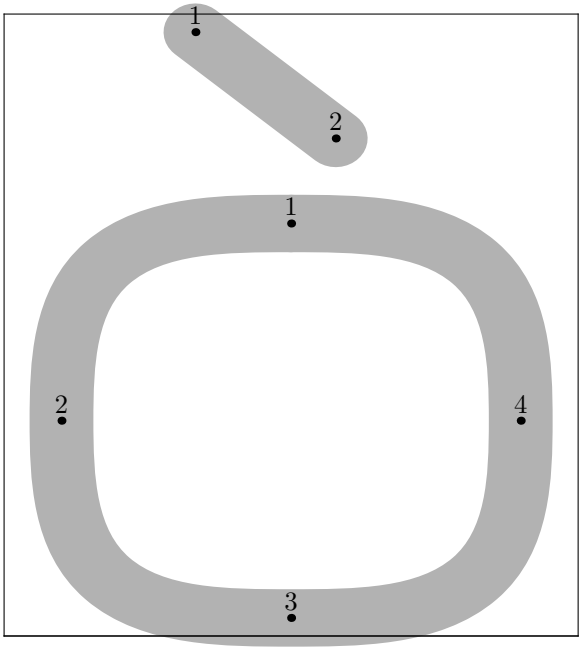
Glyph with coding number 240



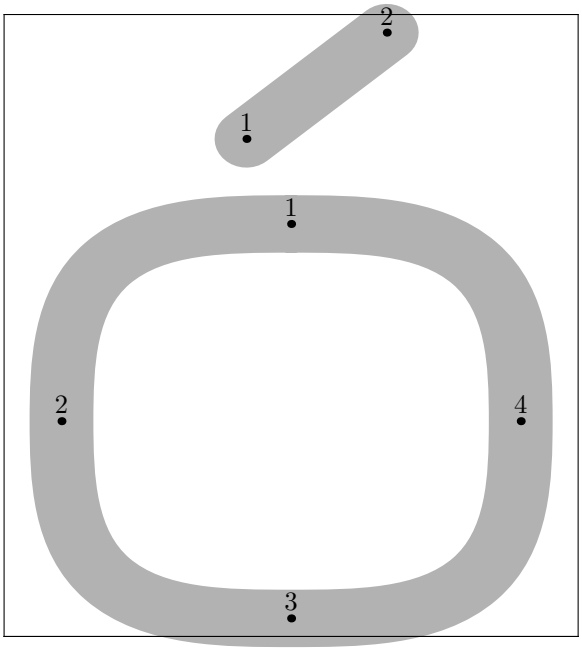
Glyph with coding number 241



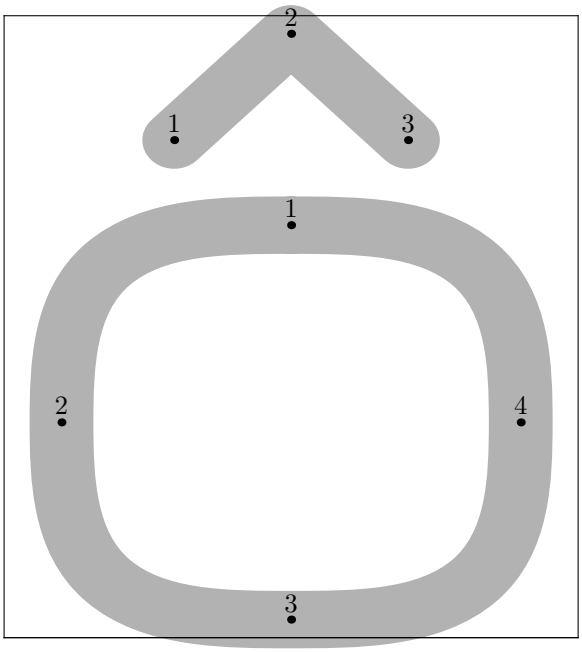
Glyph with coding number 242



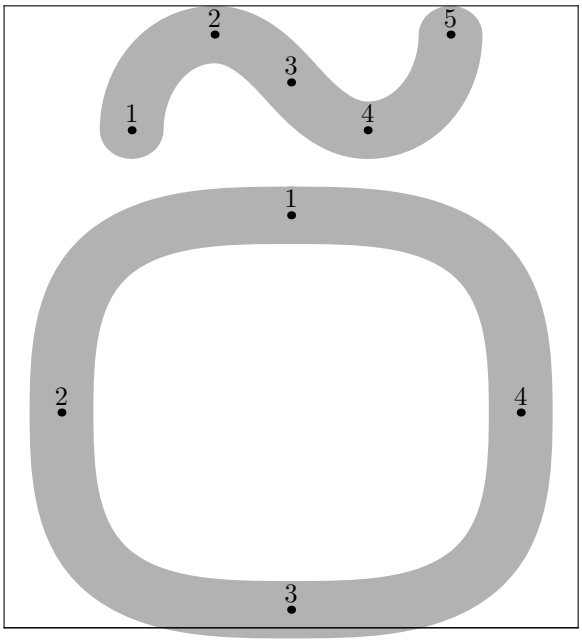
Glyph with coding number 243



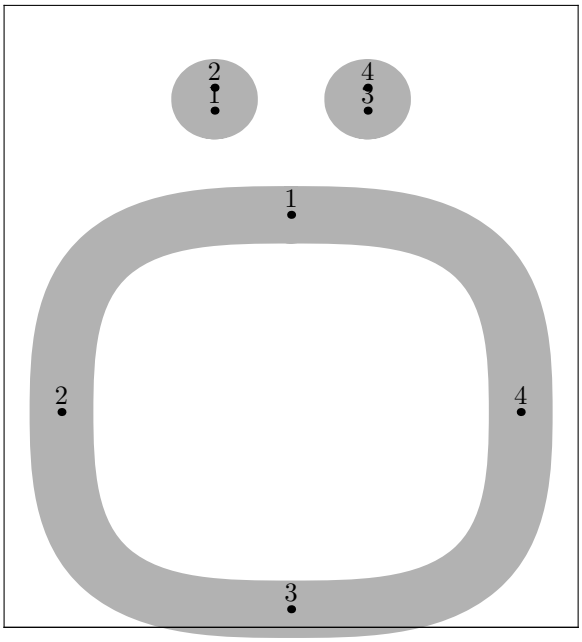
Glyph with coding number 244



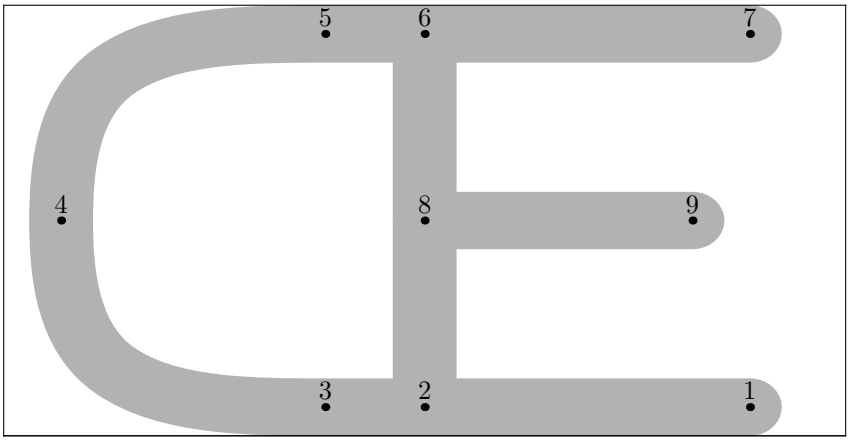
Glyph with coding number 245



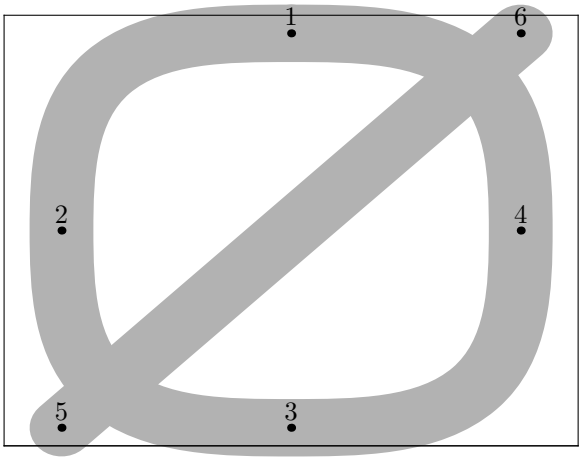
Glyph with coding number 246



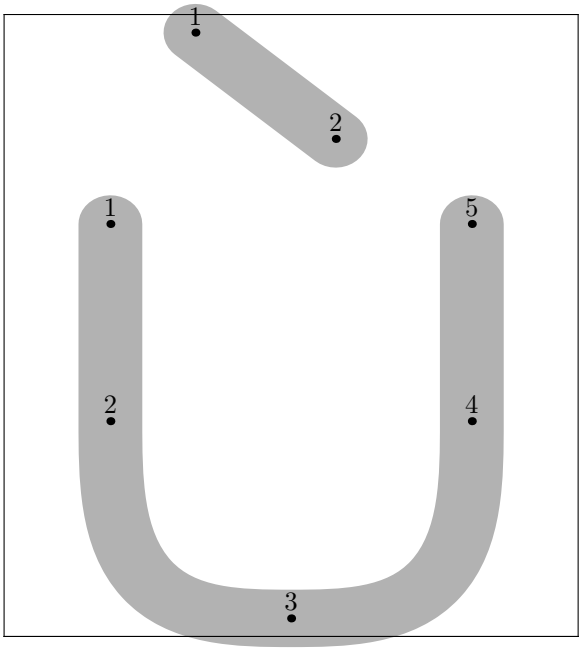
Glyph with coding number 247



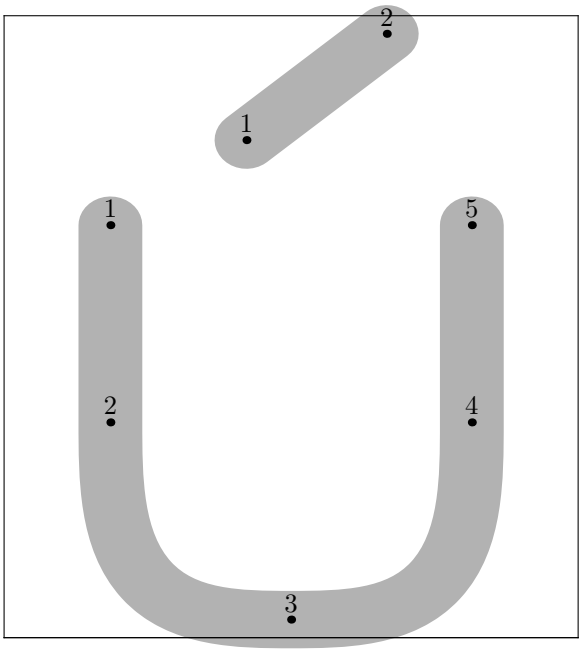
Glyph with coding number 248



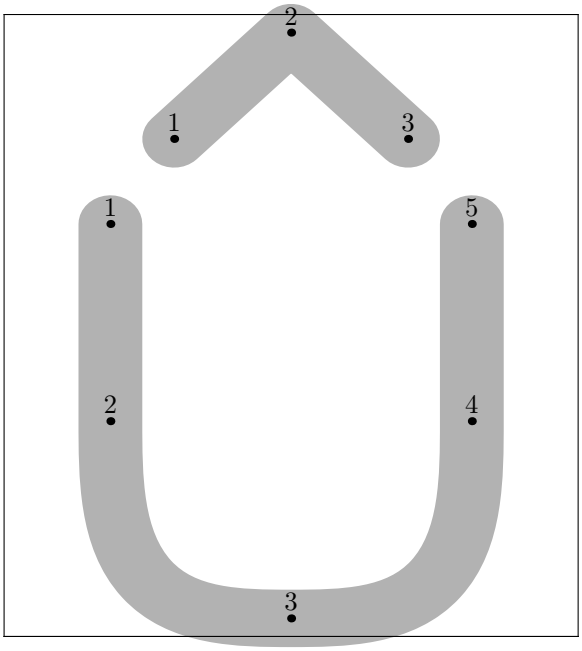
Glyph with coding number 249



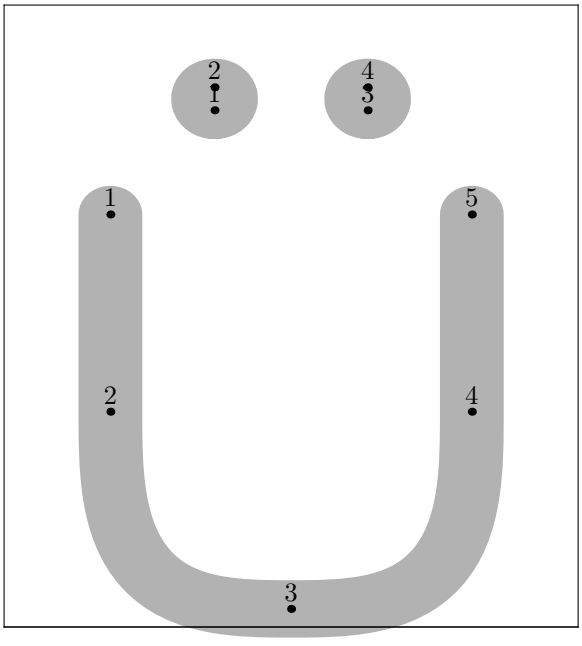
Glyph with coding number 250



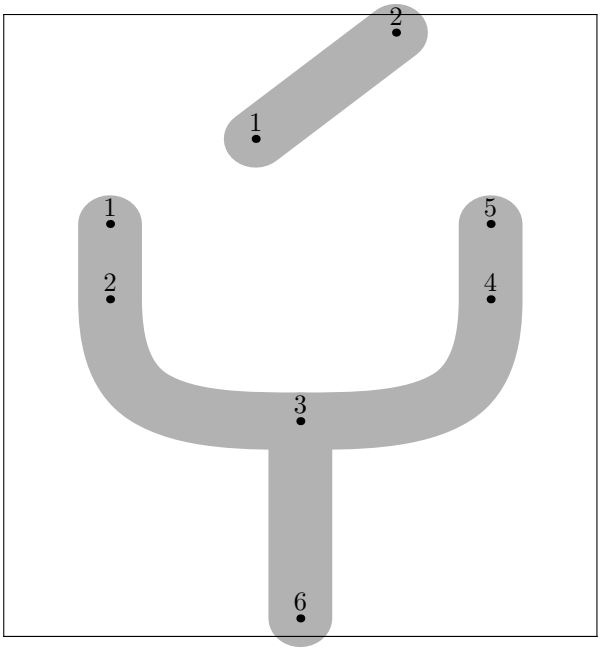
Glyph with coding number 251



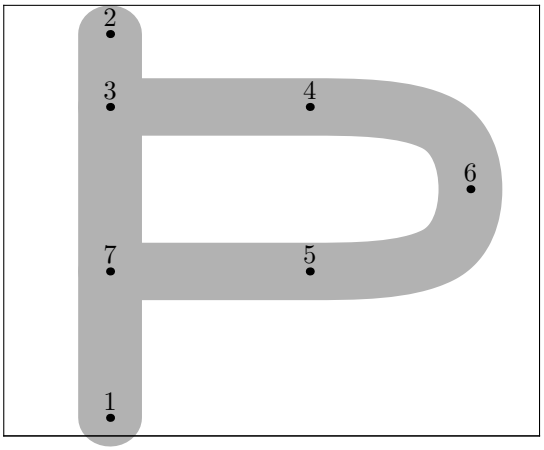
Glyph with coding number 252



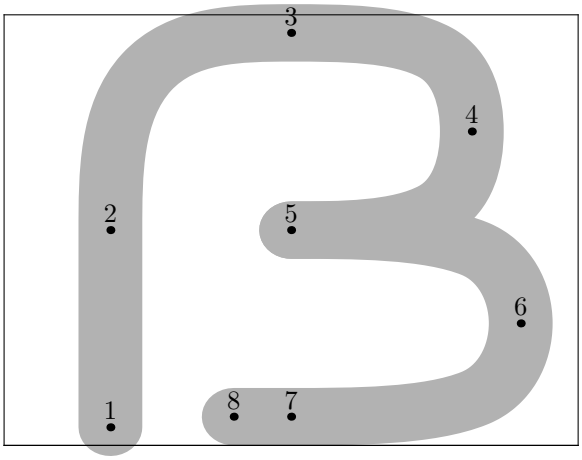
Glyph with coding number 253



Glyph with coding number 254



Glyph with coding number 255



6 Definition Of All Glyphs

% uppercase letters

ffmchar("A", 15, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_3 = .5w + noise$;
$w - x_4 = leftstemloc + noise$;
$w - x_5 = leftstemloc + noise$;
$bot\ y_1 = noise - o$;
$y_2 = barheight + noise$;
$top\ y_3 = h + o$; %no noise because of Aring
$y_4 = barheight + noise$;
$bot\ y_5 = noise - o$;
$bot\ y_6 = 0$; %no noise
$z_6 = whatever[z_4, z_5]$;
draw z_1 -- z_2 -- z_4 -- z_5;
draw $half(z_2, z_2 - z_1, z_3, randrt, z_4, z_5 - z_4)$;
$charanchortops\_[charcode] = (.5w, h)$;
$charanchorbots\_[charcode] = z_6$;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("AE", 22, $ht^{\#}$, 0);
italcorr $.9ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_3 = .5w + noise$;
$x_4 = .5w + noise$;
$x_5 = w - leftstemloc + o + noise$;
$x_6 = w - leftstemloc + o - xgap + noise$;
$x_7 = w - leftstemloc + o + noise$;
$bot\ y_1 = noise - o$;
$y_2 = barheight + noise$;
$top\ y_3 = h + noise$;
$bot\ y_4 = noise$;
$top\ y_5 = h + noise$;
$y_6 = barheight + noise$;
$bot\ y_7 = noise$;
draw z_1 -- z_2 -- z_6;
draw $arc(z_2, z_2 - z_1, z_3, z_5 - z_3)$;
draw z_5 -- z_3 -- z_4 -- z_7;
labels(1, 2, 3, 4, 5, 6, 7);

endchar;

ffmchar("B", 14, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_3 = leftstemloc + noise$;
$x_7 = .5[w - x_1, lft\ w] + noise$;
$x_5 = .85[x_1, x_7] + noise$;
$x_8 = .5[x_1, x_5] + noise$;
$x_4 = .4[x_1, x_5] + noise$;
$x_6 = .4[x_1, x_5] + noise$;
$bot\ y_1 = noise$;
$y_2 = barheight + noise$;
$top\ y_3 = h + noise$;
$y_4 = y_3 + noise$;
$y_6 = y_2 + noise$;
$bot\ y_8 = noise$;
$y_5 = .5[y_4, y_6] + noise$;
$y_7 = .5[y_6, y_8] + noise$;
$z_2 = whatever[z_1, z_3]$;
$z_9 = whatever[z_2, z_6]$;
draw z_1 -- z_3 -- z_4
  & $half(z_4, z_4 - z_3, z_5, -randup, z_6, z_2 - z_6)$
  & z_6 -- z_2;
draw $half(z_6, z_6 - z_2, z_7, -randup, z_8, z_1 - z_8)$
  & z_8 -- z_1;
labels(1, 2, 3, 4, 5, 6, 7, 8);
endchar;

ffmchar("C", 14, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = w - leftstemloc + ho + noise$;
$x_2 = .55w + noise$;
$x_3 = good.x(1.5u + s + noise)$;
$x_4 = .55w + noise$;
$x_5 = w - leftstemloc + ho + noise$;
$top\ y_1 = h + noise$;
$top\ y_2 = h + noise$;
$y_3 = barheight + noise$;
$bot\ y_4 = 0$; %no noise!
$z_5 = z_4 + whatever * randrt$;
draw z_1 -- z_2
  & $half(z_2, z_2 - z_1, z_3, -randup, z_4, z_5 - z_4)$
  & z_4 -- z_5;
$charanchortops\_[charcode] = (.5w, h)$;

*charanchorbots_*[*charcode*] = z_4;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("D", 14, $ht^{\#}$, 0);
italcorr $.9ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_3 = .45w + noise$;
$x_5 = .45w + noise$;
$w - x_4 = good.x(1.5u + s + noise)$;
bot $y_1 = noise$;
bot $y_5 = noise$;
top $y_2 = h + noise$;
top $y_3 = h + noise$;
$y_4 = barheight + noise$;
draw z_1 -- z_2 -- z_3
  & *half*(z_3, $z_3 - z_2$, z_4, $-randup$, z_5, $z_1 - z_5$)
  & z_5 -- cycle;
*charanchortops_*[*charcode*] = $(.5w, h)$;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("Eth", 14, $ht^{\#}$, 0);
italcorr $.9ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_3 = .5w + noise$;
$x_5 = .5w + noise$;
$w - x_4 = good.x(1.5u + s + noise)$;
$x_6 = eps + noise$;
$x_7 = .5w + noise$;
bot $y_1 = noise$;
bot $y_5 = noise$;
top $y_2 = h + noise$;
top $y_3 = h + noise$;
$y_4 = barheight + noise$;
$y_6 = barheight + noise$;
$y_7 = barheight + noise$;
draw z_1 -- z_2 -- z_3
  & *half*(z_3, $z_3 - z_2$, z_4, $-randup$, z_5, $z_1 - z_5$)
  & z_5 -- cycle;
draw z_6 -- z_7;
*charanchortops_*[*charcode*] = $(.5w, h)$;
labels(1, 2, 3, 4, 5, 6, 7);

endchar;

ffmchar("E", 14, $ht^{\#}$, 0);
italcorr $.9ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_3 = leftstemloc + noise$;
$x_4 = w - leftstemloc + ho + noise$;
$x_5 = w - leftstemloc + ho - xgap + noise$;
$x_6 = w - leftstemloc + ho + noise$;
$x_7 = .9[x_1, x_6]$;
bot $y_1 = noise$;
$y_2 = barheight + noise$;
top $y_3 = h + noise$;
top $y_4 = h + noise$;
$y_5 = barheight + noise$;
bot $y_7 = 0$;
$z_2 = whatever[z_1, z_3]$;
$z_6 = whatever[z_1, z_7]$;
draw z_6 -- z_1 -- z_2 -- z_5;
draw z_2 -- z_3 -- z_4;
$charanchortops\_[charcode] = (.5[leftstemloc, w - leftstemloc + o], h)$;
$charanchorbots\_[charcode] = z_7$;
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar("F", 14, $ht^{\#}$, 0);
italcorr $.9ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_3 = leftstemloc + noise$;
$x_4 = w - leftstemloc + ho + noise$;
$x_5 = w - leftstemloc + ho - xgap + noise$;
bot $y_1 = noise - o$;
$y_2 = barheight + noise$;
top $y_3 = h + noise$;
top $y_4 = h + noise$;
$y_5 = barheight + noise$;
bot $y_6 = noise$;
$z_2 = whatever[z_1, z_3]$;
draw z_1 -- z_2 -- z_5;
draw z_2 -- z_3 -- z_4;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("G", 14, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;

$x_3 = good.x(1.5u + s + noise)$;
$x_1 = w - leftstemloc + noise$;
$x_5 = w - leftstemloc + noise$;
$x_6 = w - leftstemloc + noise$;
$x_2 = .55w + noise$;
$x_4 = .55w + noise$;
$x_7 = .55w + noise$;
$y_3 = barheight + noise$;
$y_6 = barheight + noise$;
$y_7 = barheight + noise$;
$top\ y_2 = h + noise$;
$top\ y_1 = h + noise$;
$bot\ y_4 = noise$;
$bot\ y_5 = noise$;
draw z_1 -- z_2
  & $half(z_2, z_2 - z_1, z_3, -randup, z_4, z_5 - z_4)$
  & z_4 -- z_5 -- z_6 -- z_7;
$charanchortops\_[charcode] = (.5w, h)$;
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar("H", 15, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$w - x_3 = leftstemloc + noise$;
$w - x_4 = leftstemloc + noise$;
$bot\ y_1 = noise - o$;
$top\ y_2 = h + o + noise$;
$bot\ y_3 = noise - o$;
$top\ y_4 = h + o + noise$;
$y_5 = barheight + noise$;
$y_6 = barheight + noise$;
$z_5 = whatever[z_1, z_2]$;
$z_6 = whatever[z_3, z_4]$;
draw z_1 -- z_5 -- z_6 -- z_3;
draw z_5 -- z_2;
draw z_6 -- z_4;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("I", 6, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = .5w + noise$;
$x_2 = .5w + noise$;

bot $y_1 = noise - o$;
top $y_2 = h + o + noise$;
draw z_1 -- z_2;
$charanchortops\_[charcode] = (.5w + noise, h)$;
labels(1, 2);
endchar;

ffmchar("J", 9, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
lft $x_1 = noise - eps$;
$x_2 = x_1 + .5u$;
$w - x_3 = leftstemloc + noise$;
$w - x_4 = leftstemloc + noise$;
bot $y_1 = noise - o$;
$y_3 = barheight + noise$;
top $y_4 = h + noise$;
$z_2 = z_1 + whatever * randrt$;
draw z_1 -- z_2
  & $arc(z_2, randrt, z_3, z_4 - z_3)$
  & z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("K", 13, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$w - x_5 = good.x(1.5u + s + noise)$;
$w - x_6 = good.x(1.5u + s + noise)$;
bot $y_1 = noise - o$;
bot $y_6 = noise - o$;
top $y_2 = h + o + noise$;
top $y_5 = h + o + noise$;
$y_3 = .618[y_2, y_1] + noise$;
$z_3 = whatever[z_1, z_2]$;
$z_4 = whatever[z_3, z_5] = whatever[z_2, z_6]$;
draw z_1 -- z_2;
draw z_3 -- z_5;
draw z_4 -- z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("L", 12, $ht^{\#}$, 0);
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;

$rt\ x_3 = w - eps + noise$;
$bot\ y_1 = noise$;
$bot\ y_3 = noise$;
$top\ y_2 = h + noise$;
draw z_3 -- z_1 -- z_2;
$charanchortops\_[charcode] = (leftstemloc, h)$;
$charanchortoprights\_[charcode] = (.618w, h)$;
labels(1, 2, 3);
endchar;

ffmchar(`"Lslash"`, 13, $ht^{\#}$, 0);
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$w - x_3 = leftstemloc - ho + noise$;
$lft\ x_4 = eps + noise$;
$x_5 = .45w + noise$;
$bot\ y_1 = noise$;
$bot\ y_3 = noise$;
$top\ y_2 = h + noise$;
$y_4 = .4h + noise$;
$z_5 = z_4 + whatever * \mathrm{dir}(40)$;
draw z_3 -- z_1 -- z_2;
draw z_4 -- z_5;
$charanchortops\_[charcode] = (leftstemloc, h)$;
$charanchortoprights\_[charcode] = (.5w, h)$;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar(`"M"`, 18, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_3 = .5w + noise$;
$x_4 = w - leftstemloc + noise$;
$x_5 = w - leftstemloc + noise$;
$bot\ y_1 = noise - o$;
$top\ y_2 = h + o + noise$;
$bot\ y_3 = ygap - o + noise$;
$top\ y_4 = h + o + noise$;
$bot\ y_5 = noise - o$;
draw z_1 -- z_2 -- z_3 -- z_4 -- z_5;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar(`"N"`, 15, $ht^{\#}$, 0);

italcorr $.8ht\# * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_4 = w - leftstemloc + noise$;
$x_5 = w - leftstemloc + noise$;
$bot\ y_1 = noise - o$;
$top\ y_2 = h + o + noise$;
$y_3 = y_4 + ygap + noise$;
$bot\ y_4 = noise - o$;
$top\ y_5 = h + o + noise$;
$z_3 = whatever[z_4, z_5]$;
draw z_1 -- z_2 -- z_3;
draw z_4 -- z_5;
$charanchortops\_[charcode] = (.5w, h)$;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("Eng", 15, $ht\#$, $acc\_depth\#$);
italcorr $.8ht\# * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_4 = w - leftstemloc + noise$;
$x_5 = w - leftstemloc + noise$;
$x_6 = .5w + noise$;
$x_7 = .382w + noise$;
$bot\ y_1 = noise - o$;
$top\ y_2 = h + o + noise$;
$y_3 = y_4 + ygap + noise$;
$bot\ y_4 = noise - o$;
$top\ y_5 = h + o + noise$;
$bot\ y_6 = noise - d$;
$bot\ y_7 = noise - d$;
$z_3 = whatever[z_4, z_5]$;
draw z_1 -- z_2 -- z_3;
draw z_7 -- z_6
  & $arc(z_6, z_6 - z_7, z_4, z_5 - z_4)$
  & z_4 -- z_5;
$charanchortops\_[charcode] = (.5w, h)$;
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar("O", 15, $ht\#$, 0);
italcorr $.8ht\# * slant$;
$x_1 = .5w + noise$;
$x_2 = good.x(1.5u + s + noise)$;

$x_3 = .5w + noise$;
$w - x_4 = good.x(1.5u + s + noise)$;
$top\ y_1 = h + o + noise$;
$y_2 = barheight + noise$;
$bot\ y_3 = noise - o$;
$y_4 = barheight + noise$;
draw $full(z_1, -randrt, z_2, -randup, z_3, randrt, z_4, randup)$;
$charanchortops\_[charcode] = (.5w, h)$;
labels(1, 2, 3, 4);
endchar;

ffmchar("Oslash", 15, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = .5w + noise$;
$x_2 = good.x(1.5u + s + noise)$;
$x_3 = .5w + noise$;
$w - x_4 = good.x(1.5u + s + noise)$;
$x_5 = good.x(1.5u + s + noise)$;
$w - x_6 = good.x(1.5u + s + noise)$;
$top\ y_1 = h + o + noise$;
$y_2 = barheight + noise$;
$bot\ y_3 = noise - o$;
$y_4 = barheight + noise$;
$bot\ y_5 = noise - o$;
$top\ y_6 = h + o + noise$;
draw $full(z_1, -randrt, z_2, -randup, z_3, randrt, z_4, randup)$;
draw z_5 -- z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("OE", 22, $ht^{\#}$, 0);
italcorr $.9ht^{\#} * slant$;
$x_2 = .5w + noise$;
$x_4 = good.x(1.5u + s + noise)$;
$x_6 = .5w + noise$;
$x_7 = w - leftstemloc + o + noise$;
$x_9 = w - leftstemloc + o - xgap + noise$;
$x_1 = w - leftstemloc + o + noise$;
$x_3 = .382w + noise$;
$x_5 = .382w + noise$;
$y_4 = barheight + noise$;
$y_8 = barheight + noise$;
$top\ y_7 = h + noise$;
$y_9 = barheight + noise$;
$bot\ y_1 = noise$;

$bot\ y_3 = noise$;
$top\ y_5 = h + noise$;
$z_2 = whatever[z_1, z_3]$;
$z_6 = whatever[z_5, z_7]$;
$z_8 = whatever[z_2, z_6]$;
draw z_1 -- z_3
  & $half(z_3, z_3 - z_1, z_4, randup, z_5, z_7 - z_5)$
  & z_5 -- z_7;
draw z_2 -- z_8 -- z_9;
draw z_6 -- z_8;
labels(1, 2, 3, 4, 5, 6, 7, 8, 9);
endchar;

ffmchar("P", 14, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_3 = leftstemloc + noise$;
$x_4 = .618[x_1, w - x_1] + noise$;
$x_5 = .618[x_1, w - x_1] + noise$;
$x_6 = .5[w - x_1, lft\ w] + noise$;
$y_2 = barheight + noise$;
$y_5 = barheight + noise$;
$bot\ y_1 = noise - o$;
$top\ y_3 = h + noise$;
$top\ y_4 = h + noise$;
$y_6 = .5[y_4, y_5] + noise$;
$z_2 = whatever[z_1, z_3]$;
draw z_1 -- z_3 -- z_4
  & $half(z_4, z_4 - z_3, z_6, -randup, z_5, z_2 - z_5)$
  & z_5 -- z_2;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("Thorn", 14, $ht^{\#}$, 0);
italcorr $.7ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_4 = .618[x_1, w - x_1] + noise$;
$x_5 = .618[x_1, w - x_1] + noise$;
$x_6 = .5[w - x_1, lft\ w] + noise$;
$bot\ y_1 = noise - o$;
$top\ y_2 = h + noise$;
$y_3 = .764h + noise$;
$y_4 = .764h + noise$;
$y_6 = .5[y_4, y_5] + noise$;

$y_5 = .382h + noise$;
$y_7 = .382h + noise$;
$z_3 = whatever[z_1, z_2]$;
$z_7 = whatever[z_1, z_2]$;
draw z_1 -- z_2;
draw z_3 -- z_4
  & $half(z_4, z_4 - z_3, z_6, -randup, z_5, z_7 - z_5)$
  & z_5 -- z_7;
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar("Q", 15, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = .5w + noise$;
$x_2 = good.x(1.5u + s + noise)$;
$x_3 = .5w + noise$;
$w - x_4 = good.x(1.5u + s + noise)$;
$x_5 = \min(.618[x_4, x_3], x_4 - 2px) + noise$;
$w - x_6 = good.x(1.5u + s + noise)$;
$top\ y_1 = h + o + noise$;
$y_2 = barheight + noise$;
$bot\ y_3 = noise - o$;
$y_4 = barheight + noise$;
$y_5 = .618[y_3, y_4] + noise$;
$bot\ y_6 = noise - o$;
draw $full(z_1, -randrt, z_2, -randup, z_3, randrt, z_4, randup)$;
draw z_5 -- z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("R", 14, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_3 = leftstemloc + noise$;
$x_4 = .618[x_1, w - x_1] + noise$;
$x_6 = .618[x_1, w - x_1] + noise$;
$x_5 = .5[w - x_1, lft\ w] + noise$;
$x_7 = .5[w - x_1, lft\ w] + noise$;
$y_2 = barheight + noise$;
$y_6 = barheight + noise$;
$bot\ y_1 = noise - o$;
$bot\ y_7 = noise - o$;
$top\ y_3 = h + noise$;
$top\ y_4 = h + noise$;
$y_5 = .5[y_4, y_6] + noise$;

$z_2 = whatever[z_1, z_3]$;
draw z_1 -- z_3 -- z_4
  & $half(z_4, z_4 - z_3, z_5, -randup, z_6, z_2 - z_6)$
  & z_6 -- z_2;
draw z_6 -- z_7;
$charanchortops\_[charcode] = (.5w, h)$;
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar("S", 14, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_3 = good.x(2u + s + noise)$;
$x_8 = good.x(2u + s + noise)$;
$w - x_1 = good.x(3u + s + noise)$;
$x_2 = .382[x_3, x_1] + noise$;
$x_4 = .382[x_3, x_1] + noise$;
$x_5 = .9[x_3, x_1] + noise$;
$x_7 = .9[x_3, x_1] + noise$;
$w - x_6 = good.x(1.75u + .5s + noise)$;
$x_9 = .618[x_8, x_7]$;
$top\ y_1 = h + noise$;
$top\ y_2 = h + noise$;
$y_3 = .6[y_2, y_4] + noise$;
$y_4 = barheight + noise$;
$y_5 = barheight + noise$;
$y_6 = .3[y_5, y_7] + noise$;
$bot\ y_9 = 0$;
$z_7 = z_9 + whatever * randrt$;
$z_8 = whatever[z_7, z_9]$;
draw z_1 -- z_2
  & $half(z_2, z_2 - z_1, z_3, -randup, z_4, z_5 - z_4)$
  & z_4 -- z_5
  & $half(z_5, z_5 - z_4, z_6, -randup, z_7, z_8 - z_7)$
  & z_7 -- z_8;
$charanchortops\_[charcode] = (.5w, h)$;
$charanchorbots\_[charcode] = z_9$;
labels(1, 2, 3, 4, 5, 6, 7, 8, 9);
endchar;

ffmchar("Germandbls", 15, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_3 = .5w + noise$;
$x_5 = .5w + noise$;

$x_7 = .5w + noise$;
$w - x_4 = leftstemloc + noise$;
$w - x_6 = good.x(1.5u + s + noise)$;
$x_8 = .4w + noise$;
$bot\ y_1 = noise - o$;
$bot\ y_8 = noise$;
$y_2 = barheight + noise$;
$top\ y_3 = h + o + noise$;
$y_5 = barheight + noise$;
$y_4 = .5[y_5, y_3] + noise$;
$y_6 = .5[y_7, y_5] + noise$;
$z_7 = z_8 + whatever * randrt$;
pair *randira*, *randirb*;
$randira := randrt$;
$randirb := randrt$;
draw z_1 -- z_2
  & $arc(z_2, randup, z_3, randira)$
  & $half(z_3, randira, z_4, -randup, z_5, -randirb)$;
draw $half(z_5, randirb, z_6, -randup, z_7, z_8 - z_7)$
  & z_7 -- z_8;
labels(1, 2, 3, 4, 5, 6, 7, 8);
endchar;

ffmchar("T", 13, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant + .5u^{\#}$;
if $.5w \neq good.x\ .5w$: *change_width*; **fi**
$lft\ x_1 = noise - eps$;
$rt\ x_2 = w + noise$;
$x_3 = .5w + noise$;
$x_4 = .5w + noise$;
$top\ y_1 = h + noise$;
$top\ y_2 = h + noise$;
$bot\ y_4 = noise - o$;
$z_3 = whatever[z_1, z_2]$;
draw z_1 -- z_2;
draw z_3 -- z_4;
$charanchortops\_[charcode] = (.5w, h)$;
$charanchorbots\_[charcode] = (x_4, 0)$;
labels(1, 2, 3, 4);
endchar;

ffmchar("Tcedilla", 13, $ht^{\#}$, $acc\_depth^{\#}$);
italcorr $ht^{\#} * slant + .5u^{\#}$;
if $.5w \neq good.x\ .5w$: *change_width*; **fi**
$lft\ x_1 = noise - eps$;

$rt\ x_2 = w + noise$;
$x_3 = .5w + noise$;
$x_4 = .5w + noise$;
$x_5 = .55w + noise$;
$x_6 = .45w + noise$;
$top\ y_1 = h + noise$;
$top\ y_2 = h + noise$;
$bot\ y_4 = noise - o$;
$top\ y_5 = noise - .2d$;
$bot\ y_6 = noise - d$;
$z_3 = whatever[z_1, z_2]$;
draw z_1 -- z_2;
draw z_3 -- z_4;
draw z_5 -- z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("U", 15, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$w - x_4 = leftstemloc + noise$;
$w - x_5 = leftstemloc + noise$;
$x_3 = .5[x_1, x_4] + noise$;
$top\ y_1 = h + o + noise$;
$y_2 = barheight + noise$;
$bot\ y_3 = noise - o$;
$y_4 = barheight + noise$;
$top\ y_5 = h + o + noise$;
draw z_1 -- z_2
  & $half(z_2, z_2 - z_1, z_3, randrt, z_4, z_5 - z_4)$
  & z_4 -- z_5;
$charanchortops\_[charcode] = (x_3, h)$;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("V", 13, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = good.x(1.5u + s + noise) - ho$;
$w - x_3 = good.x(1.5u + s + noise) - ho$;
$x_2 = .5[x_1, x_3] + noise$;
$top\ y_1 = h + o + noise$;
$bot\ y_2 = noise - o$;
$top\ y_3 = h + o + noise$;
draw z_1 -- z_2 -- z_3;

labels(1, 2, 3);
endchar;

ffmchar("W", 20, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = good.x(1.5u + s + noise) - ho$;
$w - x_5 = good.x(1.5u + s + noise) - ho$;
$x_3 = .5[x_1, x_5] + noise$;
$top\ y_1 = h + o + noise$;
$bot\ y_2 = noise - o$;
$y_3 = y_1 - ygap + noise$;
$bot\ y_4 = noise - o$;
$top\ y_5 = h + o + noise$;
$z_4 = z_5 + whatever * (x_5 - x_1, 4 * (y_1 - y_2) - 2ygap)$;
$z_2 = z_1 + whatever * (x_1 - x_5, 4 * (y_1 - y_2) - 2ygap)$;
draw z_1 -- z_2 -- z_3;
draw z_3 -- z_4 -- z_5;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("X", 13, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = good.x(1.5u + s + noise)$;
$x_3 = good.x(1.5u + s + noise)$;
$w - x_2 = good.x(1.5u + s + noise)$;
$w - x_4 = good.x(1.5u + s + noise)$;
$top\ y_1 = h + o + noise$;
$top\ y_4 = h + o + noise$;
$bot\ y_3 = noise - o$;
$bot\ y_2 = noise - o$;
draw z_1 -- z_2;
draw z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("Y", 15.5, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + .5noise$;
$w - x_4 = leftstemloc + .5noise$;
$w - x_5 = leftstemloc + noise$;
$x_3 = .5[x_1, x_5] + noise$;
$x_6 = .5[x_1, x_5] + noise$;
$top\ y_1 = h + o + noise$;
$top\ y_5 = h + o + noise$;

bot $y_6 = noise - o$;
$y_3 = barheight + noise$;
$y_2 = .618[y_3, y_5] + noise$;
$y_4 = .618[y_3, y_5] + noise$;
draw z_1 -- z_2
  & *half*(z_2, $z_2 - z_1$, z_3, *randrt*, z_4, $z_5 - z_4$)
  & z_4 -- z_5;
draw z_6 -- z_3;
*charanchortops_*[*charcode*] = $(.5w, h)$;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("Z", 15, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$w - x_2 = leftstemloc + noise$;
$x_3 = leftstemloc + noise$;
$w - x_4 = leftstemloc + noise$;
top $y_1 = h + noise$;
top $y_2 = h + noise$;
bot $y_3 = noise$;
bot $y_4 = noise$;
draw z_1 -- z_2 -- z_3 -- z_4;
*charanchortops_*[*charcode*] = $(.5w, h)$;
labels(1, 2, 3, 4);
endchar;

% chained uppercase letters

ffmchainedchar("IJ", "I", "J");

% combined uppercase letters

ffmcombinedchar("Aacute", "A", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Agrave", "A", "grave", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Adieresis", "A", "dieresis", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Acircumflex", "A", "circumflex", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Atilde", "A", "tilde", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Aring", "A", "ring", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Abreve", "A", "breve", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Aogonek", "A", "ogonek", "bot", $ht^{\#}$, $comma\_depth^{\#}$);

ffmcombinedchar("Cacute", "C", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ccaron", "C", "caron", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ccedilla", "C", "cedilla", "bot", $ht^{\#}$, $acc\_depth^{\#}$);

ffmcombinedchar("Dcaron", "D", "caron", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Eacute", "E", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Egrave", "E", "grave", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Edieresis", "E", "dieresis", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ecircumflex", "E", "circumflex", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ecaron", "E", "caron", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Eogonek", "E", "ogonek", "bot", $ht^{\#}$, $comma\_depth^{\#}$);

ffmcombinedchar("Gbreve", "G", "breve", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Iacute", "I", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Igrave", "I", "grave", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Idieresis", "I", "dieresis", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Icircumflex", "I", "circumflex", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Lacute", "L", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Lcaron", "L", "quoteright", "topright", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Nacute", "N", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ntilde", "N", "tilde", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ncaron", "N", "caron", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Oacute", "O", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ograve", "O", "grave", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Odieresis", "O", "dieresis", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ocircumflex", "O", "circumflex", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Otilde", "O", "tilde", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ohungarumlaut", "O", "hungarumlaut", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Racute", "R", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Rcaron", "R", "caron", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Sacute", "S", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Scaron", "S", "caron", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Scedilla", "S", "cedilla", "bot", $ht^{\#}$, $comma\_depth^{\#}$);

ffmcombinedchar("Tcaron", "T", "caron", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Uacute", "U", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ugrave", "U", "grave", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Udieresis", "U", "dieresis", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ucircumflex", "U", "circumflex", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Uhungarumlaut", "U", "hungarumlaut", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Uring", "U", "ring", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Yacute", "Y", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Ydieresis", "Y", "dieresis", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Zacute", "Z", "acute", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Zcaron", "Z", "caron", "top", $ht^{\#} + acc\_ht^{\#}$, 0);

ffmcombinedchar("Zdotaccent", "Z", "dotaccent", "top", $ht^{\#} + acc\_ht^{\#}$, 0);
% accents

ffmchar("acute", 6, $x\_ht^{\#} + acc\_ht^{\#}$, 0);
lft $x_1 = .2w + noise$;
rt $x_2 = w + noise$;
bot $y_1 = .2[x\_ht, h] + noise$;
top $y_2 = h + o + noise$;
draw z_1 -- z_2;
$charanchortops\_[charcode] = (.5w, x\_ht)$;
labels(1, 2);
endchar;

ffmchar("grave", 6, $x\_ht^{\#} + acc\_ht^{\#}$, 0);
lft $x_1 = noise$;
rt $x_2 = .8w + noise$;
top $y_1 = h + o + noise$;
bot $y_2 = .2[x\_ht, h] + noise$;
draw z_1 -- z_2;
$charanchortops\_[charcode] = (.5w, x\_ht)$;
labels(1, 2);
endchar;

ffmchar("dieresis", 8, $x\_ht^{\#} + acc\_ht^{\#}$, 0);
$x_1 = x_2 = .5w - \max(1.8u, (1 + dotincr) * .7px) + noise$;
$x_3 = x_4 = .5w + \max(1.8u, (1 + dotincr) * .7px) + noise$;
bot $y_1 = .3[x\_ht, h] + noise$;
$y_2 = y_1 + dotincr * py$;
bot $y_3 = .3[x\_ht, h] + noise$;
$y_4 = y_3 + dotincr * py$;
draw $dotcircle(z_1, z_2)$;
draw $dotcircle(z_3, z_4)$;
$charanchortops\_[charcode] = (.5w, x\_ht)$;

labels(1, 2, 3, 4);
endchar;

ffmchar("circumflex", 7, $x\_ht^{\#} + acc\_ht^{\#}$, 0);
lft $x_1 = noise$;
$x_2 = .5w + noise$;
rt $x_3 = w + noise$;
bot $y_1 = .2[x\_ht, h] + noise$;
bot $y_3 = .2[x\_ht, h] + noise$;
top $y_2 = h + o + noise$;
draw z_1 -- z_2 -- z_3;
$charanchortops\_[charcode] = (.5w, x\_ht)$;
labels(1, 2, 3);
endchar;

ffmchar("tilde", 9, $x\_ht^{\#} + acc\_ht^{\#}$, 0);
lft $x_1 = eps + noise$;
$x_2 = .3w + noise$;
$x_4 = .7w + noise$;
$x_3 = .5w + noise$;
rt $x_5 = w - eps + noise$;
bot $y_1 = .2[x\_ht, h] + noise$;
bot $y_4 = .2[x\_ht, h] + noise$;
top $y_2 = h + noise$;
top $y_5 = h + noise$;
$y_3 = .6[x\_ht, h] + noise$;
if angle **direction** 1 **of**($z_2\{right\} \ldots z_3 \ldots z_4\{right\}$) < -90:
  draw $z_1\{randup\} \ldots z_2\{randrt\} \ldots z_3\{-randup\} \ldots z_4\{randrt\} \ldots z_5\{randup\}$;
else:
  draw $z_1\{randup\} \ldots z_2\{randrt\} \ldots z_3 \ldots z_4\{randrt\} \ldots z_5\{randup\}$;
fi
$charanchortops\_[charcode] = (.5w, x\_ht)$;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("hungarumlaut", 11, $x\_ht^{\#} + acc\_ht^{\#}$, 0);
$x_2 = .2w + noise$;
$x_3 = .8w + noise$;
$x_1 = .4w + noise$;
$x_4 = .6w + noise$;
top $y_1 = h + o + noise$;
top $y_3 = h + o + noise$;
bot $y_2 = .2[x\_ht, h] + noise$;
bot $y_4 = .2[x\_ht, h] + noise$;
draw z_1 -- z_2;

draw z_3 -- z_4;
*charanchortops_*[*charcode*] = (.4*w*, *x_ht*);
labels(1, 2, 3, 4);
endchar;

ffmchar("ring", 5, *x_ht*# + *acc_ht*#, 0);
lft $x_3 = -o + noise$;
rt $x_1 = w + o + noise$;
$x_2 = .5w + noise$;
$x_4 = .5w + noise$;
top $y_4 = x\_ht + o$; %no noise because of Aring
top $y_2 = h + o + noise$;
$y_1 = .5[y_2, y_4] + noise$;
$y_3 = .5[y_2, y_4] + noise$;
draw *full*(z_1, *randup*, z_2, −*randrt*, z_3, −*randup*, z_4, *randrt*);
*charanchortops_*[*charcode*] = (.5*w*, *x_ht*);
labels(1, 2, 3, 4);
endchar;

ffmchar("caron", 7, *x_ht*# + *acc_ht*#, 0);
lft $x_1 = noise$;
$x_2 = .5w + noise$;
rt $x_3 = w + noise$;
top $y_1 = h + o + noise$;
top $y_3 = h + o + noise$;
bot $y_2 = .2[x\_ht, h] + noise$;
draw z_1 -- z_2 -- z_3;
*charanchortops_*[*charcode*] = (.5*w*, *x_ht*);
labels(1, 2, 3);
endchar;

ffmchar("breve", 7, *x_ht*# + *acc_ht*#, 0);
lft $x_1 = noise$;
$x_2 = .5w + noise$;
rt $x_3 = w + noise$;
top $y_1 = h + o + noise$;
top $y_3 = h + o + noise$;
bot $y_2 = .2[x\_ht, h] + noise$;
draw *half*(z_1, −*randup*, z_2, *randrt*, z_3, *randup*);
*charanchortops_*[*charcode*] = (.5*w*, *x_ht*);
labels(1, 2, 3);
endchar;

ffmchar("macron", 6, *x_ht*# + *acc_ht*#, 0);
lft $x_1 = noise$;
rt $x_2 = w + noise$;

```
y1 = .5[x_ht, h] + noise;
y2 = .5[x_ht, h] + noise;
draw z1 -- z2;
charanchortops_[charcode] = (.5w, x_ht);
labels(1, 2);
endchar;

ffmchar("dotaccent", 4, x_ht# + acc_ht#, 0);
x1 = x2 = .5w + noise;
bot y1 = .5[x_ht, h] + noise;
y2 = y1 + dotincr * py;
draw dotcircle(z1, z2);
charanchortops_[charcode] = (.5w, x_ht);
labels(1, 2);
endchar;

ffmchar("cedilla", 4, x_ht#, acc_depth#);
x1 = .5w;                                          %no noise!
lft x2 = .2w + .5noise;
rt x3 = w + o + .5noise;
x4 = x2;
lft x5 = 0;
bot y1 = 0;                                        %no noise!
y2 = .4[y1, y4];
y3 = .7[y1, y4];
bot y4 = noise - d;
z5 = z4 + whatever * randrt;
draw z5 -- z4
   & half(z4, z4 - z5, z3, randup, z2, -randrt)
   & z2 -- z1;
charanchorbots_[charcode] = z1;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("ogonek", 4, x_ht#, acc_depth#);
x1 = .6w;                                          %no noise!
lft x2 = -o + .5noise;
rt x4 = w + .5noise;
x3 = x4 - .5u;
bot y1 = 0;                                        %no noise!
y2 = .7[y1, y4];
bot y3 = noise - d;
z4 = z3 + whatever * randrt;
pair randir;
randir := -randup;
```

draw z_1 .. tension *infinity* and 1 .. z_2{*randir*}
  & *arc*(z_2, *randir*, z_3, *randrt*)
  & z_3 -- z_4;
*charanchorbots*_[*charcode*] = z_1;
labels(1, 2, 3, 4, 5);
endchar;

% digits

ffmchar("zero", 11, $ht^{\#}$, 0);
italcorr $.6ht^{\#} * slant$;
$x_1 = .5w + noise$;
$x_2 = good.x(1.5u + s + noise)$;
$x_3 = .5w + noise$;
$w - x_4 = good.x(1.5u + s + noise)$;
top $y_1 = h + o + noise$;
$y_2 = barheight + noise$;
bot $y_3 = noise - o$;
$y_4 = barheight + noise$;
draw *full*(z_1, $-randrt$, z_2, $-randup$, z_3, *randrt*, z_4, *randup*);
labels(1, 2, 3, 4);
endchar;

ffmchar("one", 11, $ht^{\#}$, 0);
italcorr $.7ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$w - x_2 = good.x(4.5u + s + noise)$;
$w - x_3 = good.x(4.5u + s + noise)$;
$y_1 = .618h + noise$;
top $y_2 = h + o + noise$;
bot $y_3 = noise - o$;
draw z_1 -- z_2 -- z_3;
labels(1, 2, 3);
endchar;

ffmchar("two", 11, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 - ho = good.x(1.5u + s + noise)$;
$x_5 = good.x(1.5u + s + noise)$;
$x_2 = .5[x_1, x_6] + noise$;
$w - x_3 = good.x(2u + s + noise)$;
$x_4 = .5[x_1, x_6] + noise$;
$x_6 = x_3 + ho + noise$;
top $y_2 = h + noise$;
$y_3 = .5[y_4, y_2] + noise$;
$y_4 = barheight + noise$;

$bot\ y_5 = noise$;
$bot\ y_6 = noise$;
$z_1 = z_2 + whatever * randrt$;
pair *randir*;
$randir := -randrt$;
draw z_1 `--` z_2
  & $half(z_2, z_2 - z_1, z_3, -randup, z_4, randir)$
  & $arc(z_4, randir, z_5, -randup)$
  & z_5 `--` z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar(`"three"`, 11, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 - ho = good.x(1.5u + s + noise)$;
$x_8 - ho = good.x(1.5u + s + noise)$;
$w - x_3 = good.x(2u + s + noise)$;
$w - x_6 = good.x(2.5u + s + noise)$;
$x_5 = .618[x_1, x_2] + noise$;
$x_2 = .55w + noise$;
$x_4 = .55w + noise$;
$x_7 = .55w + noise$;
$bot\ y_1 = noise$;
$bot\ y_2 = noise$;
$top\ y_7 = h + noise$;
$top\ y_8 = h + noise$;
$y_4 = barheight + .5noise$;
$y_5 = barheight + .5noise$;
$y_3 = .5[y_2, y_4] + noise$;
$y_6 = .5[y_4, y_7] + noise$;
draw z_1 `--` z_2
  & $half(z_2, z_2 - z_1, z_3, randup, z_4, z_5 - z_4)$
  & z_4 `--` z_5;
draw $half(z_4, z_4 - z_5, z_6, randup, z_7, z_8 - z_7)$
  & z_7 `--` z_8;
labels(1, 2, 3, 4, 5, 6, 7, 8);
endchar;

ffmchar(`"four"`, 11, $ht^{\#}$, 0);
italcorr $.7ht^{\#} * slant$;
$lft\ x_2 = eps + noise$;
$w - x_3 = good.x(1.5u + s + noise)$;
$w - x_4 = good.x(3.75u + s + noise)$;
$w - x_5 = good.x(3.75u + s + noise)$;
$rt\ x_1 = lft\ x_4 + noise$;

$y_4 = .618h + noise$;
$top\ y_1 = h + o + noise$;
$bot\ y_5 = noise - o$;
$y_2 = .618[y_4, y_5] + noise$;
$y_3 = .618[y_4, y_5] + noise$;
draw z_1 `--` z_2 `--` z_3;
draw z_4 `--` z_5;
labels(1, 2, 3, 4, 5);
endchar;

ffmchar(`"five"`, 11, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_5 = leftstemloc + noise$;
$x_6 = leftstemloc + noise$;
$x_7 = w - x_5 + noise$;
$x_1 = x_5 - ho + noise$;
$w - x_3 = good.x(2u + s + noise)$;
$x_2 = .5[x_5, x_3] + noise$;
$x_4 = .5[x_5, x_3] + noise$;
$bot\ y_1 = noise$;
$bot\ y_2 = noise$;
$top\ y_6 = h + noise$;
$top\ y_7 = h + noise$;
$y_4 = barheight + .5noise$;
$y_5 = barheight + .5noise$;
$y_3 = .5[y_2, y_4] + noise$;
draw z_1 `--` z_2
  & $half(z_2, z_2 - z_1, z_3, randup, z_4, z_5 - z_4)$
  & z_4 `--` z_5 `--` z_6 `--` z_7;
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar(`"six"`, 11, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = .5[x_2, x_4] + noise$;
$x_2 = good.x(2u + s + noise)$;
$x_3 = .5[x_2, x_4] + noise$;
$w - x_4 = good.x(1.75u + s + noise)$;
$x_5 = .618[x_2, x_4] + noise$;
$w - x_6 = leftstemloc + noise$;
$y_1 = barheight + noise$;
$y_2 = .5[y_1, y_3] + noise$;
$bot\ y_3 = noise - o$;
$y_4 = .5[y_1, y_3] + noise$;
$top\ y_5 = h + o + noise$;

$z_6 = z_5 + whatever * randrt$;
pair *randir*;
randir := *randup*;
draw *full*(z_1, $-randrt$, z_2, $-randir$, z_3, *randrt*, z_4, *randup*);
draw *arc*(z_2, *randir*, z_5, $z_6 - z_5$)
  & z_5 -- z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("seven", 11, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
lft $x_1 = eps + noise$;
$w - x_2 = good.x(1.5u + s + noise)$;
$x_3 = .618[x_2, x_1] + noise$;
top $y_1 = h + noise$;
top $y_2 = h + noise$;
bot $y_3 = noise - o$;
draw z_1 -- z_2 -- z_3;
labels(1, 2, 3);
endchar;

ffmchar("eight", 11, $ht^{\#}$, 0);
italcorr $.7ht^{\#} * slant$;
$x_1 = .5w + noise$;
$x_3 = .5w + noise$;
$x_6 = .5w + noise$;
$x_2 = good.x(1.5u + s + noise)$;
$w - x_4 = good.x(1.5u + s + noise)$;
$x_7 = .2[x_2, x_1] + noise$;
$w - x_5 = .2[x_2, x_1] + noise$;
$y_1 = barheight + noise$;
bot $y_3 = noise - o$;
top $y_6 = h + o + noise$;
$y_2 = .5[y_1, y_3] + noise$;
$y_4 = .5[y_1, y_3] + noise$;
$y_7 = .5[y_1, y_6] + noise$;
$z_5 = z_7 + whatever * (z_4 - z_2)$;
pair *randir*;
randir := *randrt*;
draw *full*(z_1, $-randir$, z_2, $-randup$, z_3, *randrt*, z_4, *randup*);
draw *full*(z_1, *randir*, z_5, *randup*, z_6, $-randrt$, z_7, $-randup$);
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar("nine", 11, $ht^{\#}$, 0);

italcorr $.7ht^{\#} * slant$;
$x_1 = .5[x_2, x_4] + noise$;
$x_2 = good.x(1.75u + s + noise)$;
$x_3 = .5[x_2, x_4] + noise$;
$w - x_4 = good.x(2u + s + noise)$;
$x_5 = .618[x_4, x_2] + noise$;
$x_6 = leftstemloc + noise$;
$top\ y_1 = h + o + noise$;
$y_2 = .5[y_1, y_3] + noise$;
$y_4 = .5[y_1, y_3] + noise$;
$bot\ y_5 = noise - o$;
$y_3 = barheight + noise$;
$z_6 = z_5 + whatever * randrt$;
pair *randir*;
$randir := randup$;
draw $full(z_1, -randrt, z_2, -randup, z_3, randrt, z_4, randir)$;
draw $arc(z_4, -randir, z_5, z_6 - z_5)$
  & z_5 -- z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

% punctation

ffmchar("visiblespace", 6, $ht^{\#}$, $comma\_depth^{\#}$);
$lft\ x_1 = good.x(.5u + noise)$;
$lft\ x_2 = good.x(.5u + noise)$;
$rt\ x_3 = good.x(w - .5u + noise)$;
$rt\ x_4 = good.x(w - .5u + noise)$;
$top\ y_1 = o + noise$;
$top\ y_4 = o + noise$;
$bot\ y_2 = noise - d$;
$bot\ y_3 = noise - d$;
draw z_1 -- z_2 -- z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("period", 5, $x\_ht^{\#}$, 0);
$x_1 = x_2 = .5w + noise$;
$bot\ y_1 = noise - o$;
$y_2 = y_1 + dotincr * py$;
draw $dotcircle(z_1, z_2)$;
labels(1, 2);
endchar;

ffmchar("colon", 5, $x\_ht^{\#}$, 0);
italcorr $.8barheight^{\#} * slant$;

$x_1 = x_2 = .5w + noise$;
$x_3 = x_4 = .5w + noise$;
bot $y_1 = noise - o$;
$y_2 = y_1 + dotincr * py$;
$y_3 = y_4 - dotincr * py = barheight + noise$;
draw *dotcircle*(z_1, z_2);
draw *dotcircle*(z_3, z_4);
labels(1, 2, 3, 4);
endchar;

ffmchar(`"comma"`, 6, *x_ht*$^{\#}$, *comma_depth*$^{\#}$);
$x_1 = leftstemloc$;
$x_2 = w - x_1$;
top $y_2 = .382barheight$;
bot $y_1 = -d$;
draw z_1 `--` z_2;
labels(1, 2);
endchar;

ffmchar(`"semicolon"`, 6, *x_ht*$^{\#}$, *comma_depth*$^{\#}$);
italcorr *barheight*$^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$w - x_2 = leftstemloc + noise$;
$w - x_3 = w - x_4 = leftstemloc + noise$;
$y_3 = y_4 - dotincr * py = barheight$;
top $y_2 = .382y_3$;
bot $y_1 = -d$;
draw z_1 `--` z_2;
draw *dotcircle*(z_3, z_4);
labels(1, 2, 3, 4);
endchar;

ffmchar(`"exclam"`, 5, *ht*$^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = x_2 = .5w + noise$;
$x_3 = .5w + noise$;
$x_4 = .5w + noise$;
bot $y_1 = noise - o$;
$y_2 = y_1 + dotincr * py$;
top $y_4 = h + o + noise$;
bot $y_3 = \max(.618barheight, top\ y_2 + eps) + noise$;
draw *dotcircle*(z_1, z_2);
draw z_3 `--` z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("question", 12, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_1 = good.x(1.5u + s + noise)$;
$w - x_3 = good.x(2u + s + noise)$;
$x_5 = .618[x_3, x_1] + noise$;
$x_2 = .618[x_1, x_3] + noise$;
$x_4 = .618[x_1, x_3] + noise$;
$x_6 = x_7 = .618[x_3, x_1] + noise$;
$top\ y_1 = h + noise$;
$top\ y_2 = h + noise$;
$y_4 = barheight + noise$;
$y_3 = .5[y_2, y_4] + noise$;
$bot\ y_6 = noise - o$;
$y_7 = y_6 + dotincr * py$;
$bot\ y_5 = \max(.618y_4, top\ y_7 + eps) + noise$;
pair *randir*;
$randir := -randrt$;
draw z_1 -- z_2
& $half(z_2, z_2 - z_1, z_3, -randup, z_4, randir)$
& $arc(z_4, randir, z_5, -randup)$;
draw $dotcircle(z_6, z_7)$;
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar("parenleft", 8, $ht^{\#}$, $comma\_depth^{\#}$);
italcorr $.8ht^{\#} * slant$;
$x_2 = good.x(1.5u + s + noise)$;
$w - x_1 = leftstemloc - ho + noise$;
$w - x_3 = leftstemloc - ho + noise$;
$top\ y_1 = h + o + noise$;
$bot\ y_3 = noise - o - d$;
$y_2 = .5[-d, h] + noise$;
draw $half(z_1, -randrt, z_2, -randup, z_3, randrt)$;
labels(1, 2, 3);
endchar;

ffmchar("parenright", 8, $ht^{\#}$, $comma\_depth^{\#}$);
italcorr $.8ht^{\#} * slant$;
$w - x_2 = good.x(1.5u + s + noise)$;
$x_1 = leftstemloc - ho + noise$;
$x_3 = leftstemloc - ho + noise$;
$top\ y_1 = h + o + noise$;
$bot\ y_3 = noise - o - d$;
$y_2 = .5[-d, h] + noise$;
draw $half(z_1, randrt, z_2, -randup, z_3, -randrt)$;

labels(1, 2, 3);
endchar;

ffmchar("hyphen", 6, $x\_ht^{\#}$, 0);
italcorr $.618x\_ht^{\#} * slant$;
lft $x_1 = noise$;
rt $x_2 = w + noise$;
$y_1 = .618h + noise$;
$y_2 = .618h + noise$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("emdash", 18, $x\_ht^{\#}$, 0);
italcorr $.618x\_ht^{\#} * slant$;
lft $x_1 = noise$;
rt $x_2 = w + noise$;
$y_1 = .618h + noise$;
$y_2 = .618h + noise$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("endash", 9, $x\_ht^{\#}$, 0);
italcorr $.618x\_ht^{\#} * slant$;
lft $x_1 = noise$;
rt $x_2 = w + noise$;
$y_1 = .618h + noise$;
$y_2 = .618h + noise$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("cwm", 0, $x\_ht^{\#}$, 0);
endchar;

ffmchar("quotedbl", 8, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = leftstemloc + noise$;
$x_2 = leftstemloc + noise$;
$x_3 = w - leftstemloc + noise$;
$x_4 = w - leftstemloc + noise$;
top $y_1 = h + o + noise$;
top $y_3 = h + o + noise$;
$y_2 = .5[barheight, x\_ht] + noise$;
$y_4 = .5[barheight, x\_ht] + noise$;

draw z_1 -- z_2;
draw z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("quoteleft", 5, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = .4w + noise$;
$x_2 = .6w + noise$;
$top\ y_1 = h + o + noise$;
$y_2 = .5[barheight, x\_ht] + noise$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("quoteright", 5, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = .4w + noise$;
$x_2 = .6w + noise$;
$top\ y_2 = h + o + noise$;
$y_1 = .5[barheight, x\_ht] + noise$;
draw z_1 -- z_2;
$charanchortoprights\_[charcode] = (.5w, h)$;
labels(1, 2);
endchar;

ffmchar("quotedblleft", 6, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_1 = .2w + noise$;
$x_4 = .8w + noise$;
$x_2 = .3w + noise$;
$x_3 = .7w + noise$;
$top\ y_1 = h + o + noise$;
$top\ y_3 = h + o + noise$;
$y_2 = .5[barheight, x\_ht] + noise$;
$y_4 = .5[barheight, x\_ht] + noise$;
draw z_1 -- z_2;
draw z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("quotedblright", 6, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_2 = .2w + noise$;
$x_3 = .8w + noise$;
$x_1 = .3w + noise$;

$x_4 = .7w + noise$;
$top\ y_1 = h + o + noise$;
$top\ y_3 = h + o + noise$;
$y_2 = .5[barheight, x\_ht] + noise$;
$y_4 = .5[barheight, x\_ht] + noise$;
draw z_1 -- z_2;
draw z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("quotesinglbase", 3, $x\_ht^{\#}$, $comma\_depth^{\#}$);
$x_1 = .4w + noise$;
$x_2 = .6w + noise$;
$bot\ y_1 = noise - d - o$;
$y_2 = -d + ht - .5[barheight, x\_ht] + noise$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("quotedblbase", 6, $x\_ht^{\#}$, $comma\_depth^{\#}$);
$x_2 = .3w + noise$;
$x_3 = .7w + noise$;
$x_1 = .2w + noise$;
$x_4 = .8w + noise$;
$bot\ y_1 = noise - d - o$;
$bot\ y_3 = noise - d - o$;
$y_2 = -d + ht - .5[barheight, x\_ht] + noise$;
$y_4 = -d + ht - .5[barheight, x\_ht] + noise$;
draw z_1 -- z_2;
draw z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("guilsinglleft", 7, $x\_ht^{\#}$, 0);
italcorr $x\_ht^{\#} * slant$;
$x_1 = good.x(w - 2u - s + noise)$;
$x_2 = good.x(2u + s + noise)$;
$x_3 = good.x(w - 2u - s + noise)$;
$top\ y_1 = h + o + noise$;
$bot\ y_3 = noise$;
$y_2 = .5h + noise$;
draw z_1 -- z_2 -- z_3;
labels(1, 2, 3);
endchar;

ffmchar("guilsinglright", 7, $x\_ht^{\#}$, 0);

italcorr $.7x\_ht^{\#} * slant$;
$x_1 = good.x(2u + s + noise)$;
$x_2 = good.x(w - 2u - s + noise)$;
$x_3 = good.x(2u + s + noise)$;
$top\ y_1 = h + o + noise$;
$bot\ y_3 = noise$;
$y_2 = .5h + noise$;
draw z_1 -- z_2 -- z_3;
labels(1, 2, 3);
endchar;

ffmchar("guillemotleft", 10, $x\_ht^{\#}$, 0);
italcorr $x\_ht^{\#} * slant$;
$x_1 = .4w + noise$;
$x_2 = good.x(2u + s + noise)$;
$x_3 = .4w + noise$;
$x_4 = good.x(w - 2u - s + noise)$;
$x_5 = .6w + noise$;
$x_6 = good.x(w - 2u - s + noise)$;
$top\ y_1 = h + o + noise$;
$bot\ y_3 = noise$;
$y_2 = .5h + noise$;
$top\ y_4 = h + o + noise$;
$bot\ y_6 = noise$;
$y_5 = .5h + noise$;
draw z_1 -- z_2 -- z_3;
draw z_4 -- z_5 -- z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("guillemotright", 10, $x\_ht^{\#}$, 0);
italcorr $.7x\_ht^{\#} * slant$;
$x_1 = good.x(2u + s + noise)$;
$x_2 = .4w + noise$;
$x_3 = good.x(2u + s + noise)$;
$x_4 = .6w + noise$;
$x_5 = good.x(w - 2u - s + noise)$;
$x_6 = .6w + noise$;
$top\ y_1 = h + o + noise$;
$bot\ y_3 = noise$;
$y_2 = .5h + noise$;
$top\ y_4 = h + o + noise$;
$bot\ y_6 = noise$;
$y_5 = .5h + noise$;
draw z_1 -- z_2 -- z_3;

draw z_4 -- z_5 -- z_6;
labels(1, 2, 3, 4, 5, 6);
endchar;

ffmchar("percent", 16, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_5 = good.x(1.5u + s + noise)$;
$w - x_7 = good.x(1.5u + s + noise)$;
$x_1 = .2w + noise$;
$x_2 = .8w + noise$;
$x_3 = 6.5u + s + noise$;
$x_4 = 4u + s + noise$;
$x_6 = 4u + s + noise$;
$w - x_8 = 4u + s + noise$;
$w - x_9 = 6.5u + s + noise$;
$w - x_{10} = 4u + s + noise$;
$bot\ y_1 = noise - o$;
$top\ y_2 = h + o + noise$;
$y_3 = .8h + noise$;
$top\ y_4 = h + o + noise$;
$y_5 = .8h + noise$;
$bot\ y_6 = .6h - o + noise$;
$y_7 = .2h + noise$;
$top\ y_8 = .4h + o + noise$;
$y_9 = .2h + noise$;
$bot\ y_{10} = noise - o$;
draw z_1 -- z_2;
draw $full(z_3, randup, z_4, -randrt, z_5, -randup, z_6, randrt)$;
draw $full(z_7, randup, z_8, -randrt, z_9, -randup, z_{10}, randrt)$;
labels(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
endchar;

ffmchar("perthousandzero", 8, $ht^{\#}$, 0);
$x_3 = good.x(1.5u + s + noise)$;
$w - x_1 = good.x(1.5u + s + noise)$;
$x_2 = .5w + noise$;
$x_4 = .5w + noise$;
$y_1 = .2h + noise$;
$top\ y_2 = .4h + o + noise$;
$y_3 = .2h + noise$;
$bot\ y_4 = noise - o$;
draw $full(z_1, randup, z_2, -randrt, z_3, -randup, z_4, randrt)$;
labels(1, 2, 3, 4);
endchar;

ffmchar("slash", 10, $ht^{\#}$, $comma\_depth^{\#}$);
italcorr $ht^{\#} * slant$;
$x_1 = good.x(2u + s + noise)$;
$x_2 = good.x(w - 2u - s + noise)$;
$bot\ y_1 = noise - d - o$;
$top\ y_2 = h + o + noise$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("plus", 15, $x\_ht^{\#}$, 0);
italcorr $.5x\_ht^{\#} * slant$;
$x_1 = good.x(2u + s + noise)$;
$x_2 = good.x(w - 2u - s + noise)$;
$x_3 = .5w + noise$;
$x_4 = .5w + noise$;
$y_1 = .5h + noise$;
$y_2 = .5h + noise$;
$y_3 = noise - o$;
$y_4 = h + o + noise$;
draw z_1 -- z_2;
draw z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("equal", 15, $x\_ht^{\#}$, 0);
italcorr $.8x\_ht^{\#} * slant$;
$x_1 = good.x(2u + s + noise)$;
$x_2 = good.x(w - 2u - s + noise)$;
$x_3 = good.x(2u + s + noise)$;
$x_4 = good.x(w - 2u - s + noise)$;
$y_1 = .3h + noise$;
$y_2 = .3h + noise$;
$y_3 = .8h + noise$;
$y_4 = .8h + noise$;
draw z_1 -- z_2;
draw z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("numbersign", 15, $ht^{\#}$, $comma\_depth^{\#}$);
italcorr $.8ht^{\#} * slant$;
$x_1 = good.x(2u + s + noise)$;
$x_2 = good.x(w - 2u - s + noise)$;
$x_3 = good.x(2u + s + noise)$;

$x_4 = good.x(w - 2u - s + noise)$;
$x_5 = .25w + noise$;
$x_6 = .45w + noise$;
$x_7 = .55w + noise$;
$x_8 = .75w + noise$;
$y_1 = .3x\_ht + noise$;
$y_2 = .3x\_ht + noise$;
$y_3 = .8x\_ht + noise$;
$y_4 = .8x\_ht + noise$;
bot $y_5 = 1.1x\_ht - h - o + noise$;
top $y_6 = h + o + noise$;
bot $y_7 = 1.1x\_ht - h - o + noise$;
top $y_8 = h + o + noise$;
draw z_1 -- z_2;
draw z_3 -- z_4;
draw z_5 -- z_6;
draw z_7 -- z_8;
labels(1, 2, 3, 4, 5, 6, 7, 8);
endchar;

ffmchar(`"dollar"`, 11, $ht^{\#}$, 0);
italcorr $.7ht^{\#} * slant$;
$x_3 = good.x(1.5u + s + noise)$;
$x_8 = good.x(1.5u + s + noise)$;
$w - x_1 = leftstemloc + noise$;
$w - x_6 = good.x(1.5u + s + noise)$;
$x_2 = .382[x_3, x_6] + noise$;
$x_4 = .382[x_3, x_6] + noise$;
$x_5 = .618[x_3, x_6] + noise$;
$x_7 = .618[x_3, x_6] + noise$;
$x_9 = .5w + noise$;
$x_{10} = .5w + noise$;
top $y_1 = h + noise$;
top $y_2 = h + noise$;
$y_3 = .6[y_2, y_4] + noise$;
$y_4 = barheight + noise$;
$y_5 = barheight + noise$;
$y_6 = .5[y_5, y_7] + noise$;
bot $y_7 = noise$;
bot $y_8 = noise$;
top $y_9 = noise$;
bot $y_{10} = h + noise$;
draw z_1 -- z_2
  & $half(z_2, z_2 - z_1, z_3, -randup, z_4, z_5 - z_4)$

& z_4 -- z_5
& *half*(z_5, $z_5 - z_4$, z_6, $-$*randup*, z_7, $z_8 - z_7$)
& z_7 -- z_8;
draw z_9 -- z_{10};
labels(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
endchar;

ffmchar("ampersand", 15, $ht^{\#}$, 0);
italcorr $ht^{\#} * slant$;
$x_3 = good.x(1.5u + s + noise)$;
$x_6 = good.x(1.5u + s + noise)$;
rt $x_{12} = w - eps$;
$x_1 = .618w + noise$;
$x_{11} = .618w + noise$;
$x_{10} = .5[x_{11}, x_{12}] + noise$;
$x_9 = .5[x_{11}, x_{12}] + noise$;
$x_2 = .618[x_9, x_6] + noise$;
$x_4 = .618[x_9, x_6] + noise$;
$x_7 = .618[x_9, x_6] + noise$;
$x_8 = .618[x_7, x_9] + noise$;
$x_5 = x_4 + .1u$;
top $y_1 = h + .5noise$;
top $y_2 = h + .5noise$;
bot $y_7 = .5noise$;
bot $y_8 = .5noise$;
$y_4 = barheight + noise$;
$y_{11} = barheight + noise$;
$y_{12} = barheight + noise$;
$y_3 = .5[y_2, y_4] + noise$;
$y_6 = .5[y_4, y_7] + noise$;
$y_9 = .5[y_4, y_7] + noise$;
$z_5 = z_4 + whatever * randrt$;
$z_{10} = whatever[z_{11}, z_{12}]$;
draw z_1 -- z_2
& *half*(z_2, $z_2 - z_1$, z_3, $-$*randup*, z_4, $z_5 - z_4$)
& z_4 -- z_5;
draw *half*(z_4, $z_4 - z_5$, z_6, $-$*randup*, z_7, $z_8 - z_7$)
& z_7 -- z_8
& *arc*(z_8, $z_8 - z_7$, z_9, $z_{10} - z_9$)
& z_9 -- z_{10};
draw z_{11} -- z_{12};
labels(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12);
endchar;

ffmchar("asterisk", 12, $ht^{\#}$, 0);

numeric *outerradius*, *innerradius*;
outerradius = .5*w* − 2*u* − *s*;
innerradius = *px*;
path *outercircle*, *innercircle*;
outercircle = (.5*w*, *h* + *o* + *noise*) .. (.5*w*, *h* − 2 ∗ *outerradius*) .. cycle;
innercircle = (.5*w*, *h* + *o* + *noise* − *outerradius* + *innerradius*)
.. (.5*w*, *h* + *o* + *noise* − *outerradius* − *innerradius*) .. cycle;
z_1 = **point** 0 **of** *outercircle* + (*noise*, *noise*);
z_2 = **point** 0 **of** *innercircle*;
z_3 = **point** .4 **of** *outercircle* + (*noise*, *noise*);
z_4 = **point** .4 **of** *innercircle*;
z_5 = **point** .8 **of** *outercircle* + (*noise*, *noise*);
z_6 = **point** .8 **of** *innercircle*;
z_7 = **point** 1.2 **of** *outercircle* + (*noise*, *noise*);
z_8 = **point** 1.2 **of** *innercircle*;
z_9 = **point** 1.6 **of** *outercircle* + (*noise*, *noise*);
z_{10} = **point** 1.6 **of** *innercircle*;
draw z_1 -- z_2;
draw z_3 -- z_4;
draw z_5 -- z_6;
draw z_7 -- z_8;
draw z_9 -- z_{10};
labels(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
endchar;

ffmchar("`less`", 12, *x_ht*#, 0);
italcorr *x_ht*# ∗ *slant*;
x_1 = *good.x*(*w* − 2*u* − *s* + *noise*);
x_2 = *good.x*(2*u* + *s* + *noise*);
x_3 = *good.x*(*w* − 2*u* − *s* + *noise*);
top y_1 = *h* + *o* + *noise*;
bot y_3 = *noise*;
y_2 = .5*h* + *noise*;
draw z_1 -- z_2 -- z_3;
labels(1, 2, 3);
endchar;

ffmchar("`greater`", 12, *x_ht*#, 0);
italcorr .5*x_ht*# ∗ *slant*;
x_1 = *good.x*(2*u* + *s* + *noise*);
x_2 = *good.x*(*w* − 2*u* − *s* + *noise*);
x_3 = *good.x*(2*u* + *s* + *noise*);
top y_1 = *h* + *o* + *noise*;
bot y_3 = *noise*;
y_2 = .5*h* + *noise*;

draw z_1 -- z_2 -- z_3;
labels(1, 2, 3);
endchar;

ffmchar("at", 16, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_{10} = good.x(1.5u + s + noise)$;
$w - x_8 = good.x(1.5u + s + noise)$;
$w - x_7 = 1.5u + s + noise$;
$x_2 = .5w + noise$;
$x_4 = .5w + .5noise$;
$x_9 = .5w + noise$;
$x_{11} = .5w + noise$;
$x_1 = .65w + noise$;
$x_5 = .65w + .5noise$;
$x_3 = .35w + noise$;
$x_6 = .5[x_1, x_7]$;
$top\ y_9 = h + o + noise$;
$bot\ y_{11} = noise - o$;
$y_8 = .5h + noise$;
$y_{10} = .5h + noise$;
$y_3 = .5h + noise$;
$y_1 = .33h + noise$;
$y_2 = .33h + noise$;
$y_7 = .33h + noise$;
$y_4 = .67h + .5noise$;
$y_5 = .67h + .5noise$;
$y_6 = .15h + noise$;
pair *randir*;
$randir = -randup$;
draw z_1 -- z_2
  & $half(z_2, z_2 - z_1, z_3, randup, z_4, z_5 - z_4)$
  & z_4 -- z_5 -- z_1
  & $half(z_1, z_1 - z_5, z_6, randrt, z_7, z_8 - z_7)$
  & z_7 -- z_8
  & $half(z_8, z_8 - z_7, z_9, -randrt, z_{10}, randir)$
  & $arc(z_{10}, randir, z_{11}, randrt)$;
labels(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11);
endchar;

ffmchar("bracketleft", 8, $ht^{\#}$, $comma\_depth^{\#}$);
italcorr $ht^{\#} * slant$;
$x_2 = leftstemloc + noise$;
$x_3 = leftstemloc + noise$;
$w - x_1 = leftstemloc - ho + noise$;

$w - x_4 = leftstemloc - ho + noise$;
$top\ y_1 = h + noise$;
$top\ y_2 = h + noise$;
$bot\ y_3 = noise - d$;
$bot\ y_4 = noise - d$;
draw z_1 -- z_2 -- z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("backslash", 10, $ht^{\#}$, $comma\_depth^{\#}$);
$x_1 = good.x(2u + s + noise)$;
$x_2 = good.x(w - 2u - s + noise)$;
$bot\ y_2 = noise - d - o$;
$top\ y_1 = h + o + noise$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("bracketright", 8, $ht^{\#}$, $comma\_depth^{\#}$);
italcorr $ht^{\#} * slant$;
$w - x_2 = leftstemloc + noise$;
$w - x_3 = leftstemloc + noise$;
$x_1 = leftstemloc - ho + noise$;
$x_4 = leftstemloc - ho + noise$;
$top\ y_1 = h + noise$;
$top\ y_2 = h + noise$;
$bot\ y_3 = noise - d$;
$bot\ y_4 = noise - d$;
draw z_1 -- z_2 -- z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("asciicircum", 8, $ht^{\#}$, 0);
$lft\ x_1 = eps + noise$;
$x_2 = .5w + noise$;
$rt\ x_3 = w - eps + noise$;
$bot\ y_1 = x\_ht + noise$;
$bot\ y_3 = x\_ht + noise$;
$top\ y_2 = h + o + noise$;
draw z_1 -- z_2 -- z_3;
labels(1, 2, 3);
endchar;

ffmchar("underscore", 16, 0, $comma\_depth^{\#}$);
$x_1 = 0$;
$x_2 = w$;

bot $y_1 = -.8d$;
bot $y_2 = -.8d$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("braceleft", 8, $ht^{\#}$, *comma_depth*$^{\#}$);
italcorr $ht^{\#} * slant$;
$w - x_1 = leftstemloc - ho + noise$;
$w - x_5 = leftstemloc - ho + noise$;
$x_2 = leftstemloc + noise$;
$x_4 = leftstemloc + noise$;
lft $x_3 = eps + noise$;
top $y_1 = h + o + noise$;
bot $y_5 = noise - o - d$;
$y_3 = .5[-d, h] + noise$;
$y_2 = .75[-d, h] + noise$;
$y_4 = .25[-d, h] + noise$;
pair *randira*;
randira = *randrt*;
draw *half*(z_1, $-$*randrt*, z_2, $-$*randup*, z_3, $-$*randira*);
draw *half*(z_3, *randira*, z_4, $-$*randup*, z_5, *randrt*);
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("bar", 5, $ht^{\#}$, *comma_depth*$^{\#}$);
$x_1 = .5w + noise$;
$x_2 = .5w + noise$;
top $y_1 = h + o + noise$;
bot $y_2 = noise - d - o$;
draw z_1 -- z_2;
labels(1, 2);
endchar;

ffmchar("braceright", 8, $ht^{\#}$, *comma_depth*$^{\#}$);
italcorr $.8ht^{\#} * slant$;
$x_1 = leftstemloc - ho + noise$;
$x_5 = leftstemloc - ho + noise$;
$w - x_2 = leftstemloc + noise$;
$w - x_4 = leftstemloc + noise$;
rt $x_3 = w - eps + noise$;
top $y_1 = h + o + noise$;
bot $y_5 = noise - o - d$;
$y_3 = .5[-d, h] + noise$;
$y_2 = .75[-d, h] + noise$;

```
y4 = .25[−d, h] + noise;
pair randira;
randira = randrt;
draw half(z1, randrt, z2, −randup, z3, randira);
draw half(z3, −randira, z4, −randup, z5, −randrt);
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("asciitilde", 11, x_ht#, 0);
lft x1 = eps + noise;
x2 = .3w + noise;
x4 = .7w + noise;
x3 = .5w + noise;
rt x5 = w − eps + noise;
bot y1 = .5h + noise;
bot y4 = .5h + noise;
top y2 = h + noise;
top y5 = h + noise;
y3 = .75h + noise;
if angle direction 1 of(z2{right} ... z3 ... z4{right}) < −90:
  draw z1{randup} ... z2{randrt}
    ... z3{−randup} ... z4{randrt} ... z5{randup};
else:
  draw z1{randup} ... z2{randrt}
    ... z3 ... z4{randrt} ... z5{randup};
fi
labels(1, 2, 3, 4, 5);
endchar;

ffmchar("dash", 6, x_ht#, 0);
italcorr .618x_ht# * slant;
lft x1 = noise;
rt x2 = w + noise;
y1 = .618h + noise;
y2 = .618h + noise;
draw z1 -- z2;
labels(1, 2);
endchar;

ffmchar("section", 10, ht#, comma_depth#);
italcorr .5ht# * slant;
x1 = .5w + noise;
x2 = good.x(2u + s + noise);
x3 = .5w + noise;
w − x4 = good.x(2u + s + noise);
```

$x_5 = good.x(1.5u + s + noise)$;
$x_6 = .5w + noise$;
$w - x_7 = leftstemloc + noise$;
$w - x_8 = good.x(1.5u + s + noise)$;
$x_9 = .5w + noise$;
$x_{10} = leftstemloc + noise$;
$y_1 = .65[-d, h] + noise$;
$y_2 = .5[-d, h] + noise$;
$y_3 = .35[-d, h] + noise$;
$y_4 = .5[-d, h] + noise$;
$y_5 = .5[y_1, y_6] + noise$;
$top\ y_6 = h + noise$;
$top\ y_7 = h + noise$;
$y_8 = .5[y_4, y_9] + noise$;
$bot\ y_9 = noise - d$;
$bot\ y_{10} = noise - d$;
pair *randira*, *randirb*;
$randira = -randrt$;
$randirb = randrt$;
draw $full(z_1, randira, z_2, -randup, z_3, randirb, z_4, randup)$;
draw $half(z_1, randira, z_5, randup, z_6, z_7 - z_6)$ & z_6 -- z_7;
draw $half(z_3, randirb, z_8, -randup, z_9, z_{10} - z_9)$ & z_9 -- z_{10};
$charanchortops\_[charcode] = (.5w, h)$;
labels(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
endchar;

ffmchar("exclamdown", 5, $ht^{\#} - comma\_depth^{\#}$, $comma\_depth^{\#}$);
italcorr $.8(ht^{\#} - comma\_depth^{\#}) * slant$;
$x_1 = x_2 = .5w + noise$;
$x_3 = .5w + noise$;
$x_4 = .5w + noise$;
$top\ y_1 = h + o + noise$;
$y_2 = y_1 - dotincr * py$;
$bot\ y_4 = noise - d - o$;
$top\ y_3 = \min(h - .618barheight, bot\ y_2 - eps) + noise$;
draw $dotcircle(z_1, z_2)$;
draw z_3 -- z_4;
labels(1, 2, 3, 4);
endchar;

ffmchar("questiondown", 12, $ht^{\#} - comma\_depth^{\#}$, $comma\_depth^{\#}$);
$w - x_1 = good.x(1.5u + s + noise)$;
$x_3 = good.x(2u + s + noise)$;
$x_5 = .618[x_3, x_1] + noise$;
$x_2 = .618[x_1, x_3] + noise$;

$x_4 = .618[x_1, x_3] + noise$;
$x_6 = x_7 = .618[x_3, x_1] + noise$;
$bot\ y_1 = noise - d$;
$bot\ y_2 = noise - d$;
$top\ y_6 = h + o + noise$;
$y_7 = y_6 - dotincr * py$;
$top\ y_5 = \min(h - .618barheight,\ bot\ y_7 - eps) + noise$;
$y_4 = .8[y_2, y_5] + noise$;
$y_3 = .5[y_2, y_4] + noise$;
pair *randir*;
$randir := -randrt$;
draw $arc(z_5, -randup, z_4, randir)$
  & $half(z_4, randir, z_3, -randup, z_2, z_1 - z_2)$
  & z_2 -- z_1;
draw $dotcircle(z_6, z_7)$;
labels(1, 2, 3, 4, 5, 6, 7);
endchar;

ffmchar(`"sterling"`, 11, $ht^{\#}$, 0);
italcorr $.8ht^{\#} * slant$;
$x_5 = good.x(1.5u + s + noise)$;
$x_7 = good.x(1.5u + s + noise)$;
$rt\ x_6 = w - eps + noise$;
$x_1 = w - leftstemloc + o + noise$;
$x_8 = w - leftstemloc + o - xgap + noise$;
$x_2 = .618[x_5, x_1] + noise$;
$x_3 = .618[x_2, x_5] + noise$;
$x_4 = .618[x_2, x_5] + noise$;
$top\ y_1 = h + noise$;
$bot\ y_5 = noise$;
$bot\ y_6 = noise$;
$y_7 = barheight + noise$;
$y_8 = barheight + noise$;
$y_4 = barheight + noise$;
$y_3 = .5[barheight, h] + noise$;
$z_2 = z_1 + whatever * randir$;
draw z_1 -- z_2
  & $arc(z_2, z_2 - z_1, z_3, z_4 - z_3)$
  & z_3 -- z_4
  & $arc(z_4, z_4 - z_3, z_5, z_5 - z_6)$;
draw z_5 -- z_6;
draw z_7 -- z_8;
labels(1, 2, 3, 4, 5, 6, 7, 8);
endchar;

7 Font Tables

Fetamont Light 10

ffml10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	ı	ȷ	ff	fi	fl	ffi	ffl	
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Regular 10

ffmr10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	″	‟	«	»	–	—	
´03*	°	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Regular 9

ffmr9	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	₀	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Regular 8

ffmr8	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	‟	″	‟	«	»	–	—	
´03*	₀	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	‚	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold 10

ffmb10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	″	‟	«	»	–	—	
´03*	∘	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold 9

ffmb9	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	-	·	¸	˛	'	‹	›
´02*	"	"	„	«	»	–	—	
´03*	˚	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	·	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold 8

ffmb8	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	¸	‹	›
´02*	"	″	‟	«	»	–	—	
´03*	°	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	¸	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Heavy 10

ffmh10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	˘
´01*	ˇ	-	·	¸	˛	'	‹	›
´02*	"	"	„	«	»	–	—	
´03*	◦	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	·	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Heavy 9

ffmh9	´0	´1	´2	´3	´4	´5	´6	´7
´00*	ˋ	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	“	”	„	«	»	–	—	
´03*	∘	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	·	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Heavy 8

ffmh8	´0	´1	´2	´3	´4	´5	´6	´7
´00*	ˋ	´	ˆ	˜	¨	˝	°	˘
´01*	ˇ	¯	˙	¸	˛	‚	‹	›
´02*	“	”	„	«	»	–	—	
´03*	∘	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	·	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Light Oblique 10

ffmlo10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	″	″	„	«	»	–	—	
´03*	ı	ȷ	ff	fi	fl	ffi	ffl	
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Oblique 10

ffmo10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	-	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	ı	ȷ	ff	fi	fl	ffi	ffl	
´04*	‟	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Oblique 9

ffmo9	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	₀	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Oblique 8

ffmo8	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	˚	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold Oblique 10

ffmbo10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	-	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	ı	ȷ	ff	fi	fl	ffi	ffl	
´04*	ß	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold Oblique 9

ffmbo9	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	-	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	˚	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	˛	!	"	#	$	%	&	’
´05*	(	)	*	+	‚	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	ˆ	_
´12*	‘	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	˜	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold Oblique 8

ffmbo8	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	-	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	°	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Heavy Oblique 10

ffmho10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	◦	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	·	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	IJ	İ	đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ij	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Heavy Oblique 9

ffmho9	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	˘
´01*	˘	¯	˙	¸	˛	¸	‹	›
´02*	"	″	‟	«	»	–	—	
´03*	∘	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Heavy Oblique 8

ffmho8	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	˘
´01*	ˇ	-	·	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	˚	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	·	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Light Condensed 10

ffmlc10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	"	„	«	»	–	—	
´03*	ı	I	J	FF	FI	FL	FFI	FFL
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Condensed 10

ffmc10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	"	„	«	»	–	—	
´03*	ı	ȷ	ff	fi	fl	ffi	ffl	
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold Condensed 40

ffmbc40	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	¯	˙	¸	˛	ˌ	‹	›
´02*	‖	″	‴	«	»	–	—	
´03*	₀	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	‚	!	"	#	$	%	&	'
´05*	(	)	*	+	‚	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Light Condensed Oblique 10

ffmlco10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	-	˙	¸	˛	‚	‹	›
´02*	‖	″	„	«	»	–	—	
´03*	₀	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	|	]	^	_
´12*	'	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Condensed Oblique 10

ffmco10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	-	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	°	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	‟	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	|	]	ˆ	_
´12*	'	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold Condensed Oblique 40

ffmbco40	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	ˇ
´01*	˘	-	˙	¸	˛	‚	‹	›
´02*	"	"	„	«	»	–	—	
´03*	°	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	˽	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Light Ultracondensed 10

ffmlq10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¯	˘	˙	¨
´01*	˚	¸	˝	˛	ˇ	ı	ȷ	ff
´02*	fi	fl	ffi	ffl	ł	Ł	–	—
´03*		!	"	#	$	%	&	'
´04*	(	)	*	+	,	-	.	/
´05*	0	1	2	3	4	5	6	7
´06*	8	9	:	;	<	=	>	?
´07*	@	A	B	C	D	E	F	G
´08*	H	I	J	K	L	M	N	O
´09*	P	Q	R	S	T	U	V	W
´10*	X	Y	Z	[	\	]	^	_
´11*	`	a	b	c	d	e	f	g
´12*	h	i	j	k	l	m	n	o
´13*	p	q	r	s	t	u	v	w
´14*	x	y	z	{	|	}	~	-
´15*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´16*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´17*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´18*	Ÿ	Ź	Ž	Ż	IJ	İ	đ	ſ
´19*	ă	ą	ć	č	ď	ě	ę	ğ
´20*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´21*	ř	ś	š	ş	ť	ţ	ű	ů
´22*	ÿ	ź	ž	ż	ij	¡	¿	£
´23*	À	Á	Â	Ã	Ā	Å	Æ	Ç
´24*	È	É	Ê	Ë	Ì	Í	Î	Ï
´25*	Ð	Ñ	Ò	Ó	Ô	Õ	Ō	Œ
´26*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´27*	à	á	â	ã	ā	å	æ	ç
´28*	è	é	ê	ë	ì	í	î	ï
´29*	ð	ñ	ò	ó	ô	õ	ō	œ
´30*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Light Ultracondensed Oblique 10

ffmlqo10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	‘	’	^	“	~	”	°	ˇ
´01*	˘	˙	˚	¸	˛	˝	˛	ˇ
´02*	!	"	#	$	%	&	’	(
´03*	)	*	+	,	-	.	/	0
´04*	1	2	3	4	5	6	7	8
´05*	9	:	;	<	=	>	?	@
´06*	A	B	C	D	E	F	G	H
´07*	I	J	K	L	M	N	O	P
´08*	Q	R	S	T	U	V	W	X
´09*	Y	Z	[	\	]	^	_	‘
´10*	a	b	c	d	e	f	g	h
´11*	i	j	k	l	m	n	o	p
´12*	q	r	s	t	u	v	w	x
´13*	y	z	{	|	}	~	¨	˚
´14*	Ą	˘	Ł	¤	Ľ	Ś	§	¨
´15*	Š	Ş	Ť	Ź		Ž	Ż	°
´16*	ą	˛	ł	´	ľ	ś	ˇ	¸
´17*	š	ş	ť	ź	˝	ž	ż	Ŕ
´18*	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č
´19*	É	Ę	Ë	Ě	Í	Î	Ď	Đ
´20*	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř
´21*	Ů	Ú	Ű	Ü	Ý	Ţ	ß	ŕ
´22*	á	â	ă	ä	ĺ	ć	ç	č
´23*	é	ę	ë	ě	í	î	ď	đ
´24*	ń	ň	ó	ô	ő	ö	÷	ř
´25*	ů	ú	ű	ü	ý	ţ	˙	ß
´26*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´27*	È	É	Ê	Ë	Ì	Í	Î	Ï
´28*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×
´29*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´30*	à	á	â	ã	ä	å	æ	ç
´31*	è	é	ê	ë	ì	í	î	ï

Fetamont Light Script 10

ffmlw10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	ˋ	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	˚	¸	˛	‹	›
´02*	‟	″	‟	«	»	–	—	
´03*	₀	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Script 10

ffmw10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	ˋ	ˊ	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	ˌ	‹	›
´02*	"	"	ˮ	«	»	–	—	
´03*	₀	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	ˌ	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold Script 10

ffmbw10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	¡	‹	›
´02*	"	″	‟	«	»	–	—	
´03*	˚	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	đ	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Heavy Script 10

ffmhw10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	˙	ˊ	ˆ	˜	¨	˝	°	˘
´01*	˘	¯	˙	˛	˛	˛	‹	›
´02*	"	″	‟	«	»	–	—	
´03*	∘	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	`	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	ð	ß
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Light Script Oblique 10

ffmlwo10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	¯	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	₀	ı	ȷ	FF	FI	FL	FFI	FFL
´04*	␣	!	"	#	$	%	&	'
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	'	a	b	c	d	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Script Oblique 10

ffmwo10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	°	ˇ
´01*	˘	-	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	˚	ı	ȷ	FF	FI	FL	FFI	FFL
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	IJ	İ	Đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ij	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Bold Script Oblique 10

ffmbwo10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	`	´	ˆ	˜	¨	˝	˚	˘
´01*	ˇ	-	˙	¸	˛	‚	‹	›
´02*	"	″	„	«	»	–	—	
´03*	˚	ı	ȷ	FF	FI	FL	FFI	FFL
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

Fetamont Heavy Script Oblique 10

ffmhwo10	´0	´1	´2	´3	´4	´5	´6	´7
´00*	ˋ	´	ˆ	˜	¨	˜	˚	˘
´01*	˘	¯	˙	¸	˛	¸	‹	›
´02*	"	″	‟	«	»	–	—	
´03*	•	ı	ȷ	ff	fi	fl	ffi	ffl
´04*	␣	!	"	#	$	%	&	’
´05*	(	)	*	+	,	-	.	/
´06*	0	1	2	3	4	5	6	7
´07*	8	9	:	;	<	=	>	?
´08*	@	A	B	C	D	E	F	G
´09*	H	I	J	K	L	M	N	O
´10*	P	Q	R	S	T	U	V	W
´11*	X	Y	Z	[	\	]	^	_
´12*	‘	a	b	c	˙	e	f	g
´13*	h	i	j	k	l	m	n	o
´14*	p	q	r	s	t	u	v	w
´15*	x	y	z	{	|	}	~	-
´16*	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ
´17*	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ
´18*	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů
´19*	Ÿ	Ź	Ž	Ż	Ĳ	İ	Đ	§
´20*	ă	ą	ć	č	ď	ě	ę	ğ
´21*	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ
´22*	ř	ś	š	ş	ť	ţ	ű	ů
´23*	ÿ	ź	ž	ż	ĳ	¡	¿	£
´24*	À	Á	Â	Ã	Ä	Å	Æ	Ç
´25*	È	É	Ê	Ë	Ì	Í	Î	Ï
´26*	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ
´27*	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
´28*	à	á	â	ã	ä	å	æ	ç
´29*	è	é	ê	ë	ì	í	î	ï
´30*	ð	ñ	ò	ó	ô	õ	ö	œ
´31*	ø	ù	ú	û	ü	ý	þ	ß

References

[Hosny11] Khaled Hosny. `https://github.com/khaledhosny/punk-otf/blob/master/tools/build.py`, 2011

[Jackowski01] Bogusław Jackowski, Janusz M. Nowacki, and Piotr Strzelczyk. *META-*

TYPE1: A METAPOST-based engine for generating Type 1 fonts. ntg.nl/eurotex/JackowskiMT.pdf, 2001

[Nienhuys06] Han-Wen Nienhuys. https://github.com/hanwen/mftrace/blob/master/tfm.py, 2006

[Romer14] Linus Romer. *The Fetamont Package*. 2014