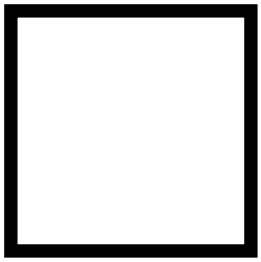
Learning PostScript by Doing

André Heck

Solutions to the Exercises

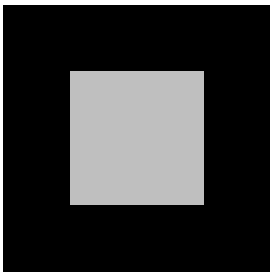
Solutions to the Exercises

EXERCISE 3



When you do not use the `closepath` operator to ensure a closed path, but replace this statement in the sample program by `0 -90 rlineto`, then line joining in the lower left corner is wrong. See the picture to the left.

EXERCISE 4



```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 100 100
% black box
newpath
   0   0 moveto
 100   0 rlineto
   0 100 rlineto
-100   0 rlineto
closepath
0 setgray
fill
% gray box
newpath
 25 25 moveto
 50  0 rlineto
  0 50 rlineto
-50  0 rlineto
closepath
0.75 setgray
fill
showpage
%EOF
```

EXERCISE 6

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 360 72
%%BeginProlog
/s 72 def
%%EndProlog
s s scale
1 s div setlinewidth

% the gray rectangle
newpath
0 0 moveto
5 0 rlineto
```

```
0 1 rlineto
-5 0 rlineto
closepath
0.8 setgray
fill

% the red arrow
newpath
 0.5 0.5  moveto
 0   0.15 rlineto
 3   0    rlineto
 0   0.25 rlineto
 1  -0.4  rlineto
-1  -0.4  rlineto
 0   0.25 rlineto
-3   0    rlineto
closepath
1 0 0 setrgbcolor
fill
showpage
%%EOF
```

EXERCISE 7

(i)

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -1 -1 99 87
%%BeginProlog
/s 28.34645 def
%%EndProlog
%%Page: 1 1
s s scale
1 s div setlinewidth

% the triangle
newpath
0 0 moveto
2 3 sqrt mul 0 lineto
3 sqrt 3 lineto
closepath
stroke

% the pedal lines
newpath
0 0 moveto
3 sqrt 1 lineto
2 3 sqrt mul 0 lineto
3 sqrt 1 moveto
3 sqrt 3 lineto
stroke
showpage
%%EOF
```

(ii)

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: -1 -1 99 87
%%BeginProlog
```

```
/s 28.34645 def
%%EndProlog
%%Page: 1 1
s s scale
1 s div setlinewidth

% the triangle
newpath
0 0 moveto
2 3 sqrt mul 0 lineto
3 sqrt 3 lineto
closepath
stroke

% the pedal lines
newpath
0 0 moveto
3 sqrt 1 lineto
2 3 sqrt mul 0 lineto
3 sqrt 1 moveto
3 sqrt 3 lineto
stroke
showpage
%%EOF
```

EXERCISE 9

```
AFPL Ghostscript 8.00 (2002-11-21)
Copyright (C) 2002 artofcode LLC, Benicia, CA. All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
GS>1 3 sqrt add 2 div 2 exp
GS<1>pstack
1.86602557
GS<1>
```

EXERCISE 10

```
AFPL Ghostscript 8.00 (2002-11-21)
Copyright (C) 2002 artofcode LLC, Benicia, CA. All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
GS>2 2 mul 1 sub
GS<1>2 dup mul 1 sub
GS<2>1 2 2 mul exch sub
GS<3>pstack
3
3
3
GS<3>
```

EXERCISE 11

```
AFPL Ghostscript 8.00 (2002-11-21)
Copyright (C) 2002 artofcode LLC, Benicia, CA. All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
GS>/cosine {
```

```
/x exch def
x 180 mul 3.14159265 div cos
} def
GS>0.5 cosine ==
0.87758255
GS>1 cosine ==
0.540302336
GS>3.14159 cosine ==
-1.0
```

Here, the introduction of the variable `x` in the program code of `cosine` is not necessary. The following definition of a cosine function using radians works as fine:

```
GS>/cosrad {180 mul 3.14159265 div cos} def
GS>30 cosrad ==
0.154249817
```

EXERCISE 14 First a straightforward version in which two arrays are taken off the stack and their sum is put back on the stack as an array.

```
AFPL Ghostscript 8.00 (2002-11-21)
Copyright (C) 2002 artofcode LLC, Benicia, CA. All rights reserved.
This software comes with NO WARRANTY: see the file PUBLIC for details.
GS>/addvec { 2 dict begin % 2 arrays on top of stack
      /u exch def
      /v exch def
      [u 0 get v 0 get add
       u 1 get v 1 get add]
   end } def
GS>/v1 [30 40] def
GS>/v2 [50 60] def
GS>v1 v2 addvec ==
[80 100]
```

The above version cannot easily be transcribed into a version without local variables. The next version can be transcribed because it first manipulates the stack so that all components of the two vectors are on top of the stack.

```
GS>/addvec { 4 dict begin % [a b] [c d] on stack
      aload pop           % [a b] c d
      3 -1 roll           % c d [a b]
      aload pop           % c d a b
      /b exch def
      /a exch def
      /d exch def
      /c exch def
      [a c add b d add]
   end } def
GS>/v1 [30 40] def
GS>/v2 [50 60] def
GS>v1 v2 addvec ==
[80 100]
```

The version without local variables looks as follows:

```
GS>/addvec {     % [a b] [c d] on stack
      aload pop  % [a b] c d
      3 -1 roll  % c d [a b]
```

```
      aload pop  % c d a b
      3 -1 roll  % c a b d
      add        % c a b+d
      3  1 roll  % b+d c a
      exch       % b+d a c
      add        % b+d a+c
      exch       % a+c b+d
      2 array astore
   } def
GS>v1 v2 addvec ==
[80 100]
```

Timing shows that the version without local variables is two times faster than the first implementation of the routine (with local variables)

EXERCISE 15

1.
```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 155 95
%%BeginProlog
/s 28.34645 def
%%EndProlog
s s scale
1 s div setlinewidth
5 s div 5 s div translate
% the triangle with vertices
% (0,0), (5,0), and (2,3)
newpath
0 0 moveto
5 0 lineto
2 3 lineto
closepath
stroke
% the medians
newpath 0 0 moveto 3.5 1.5 lineto closepath stroke
newpath 5 0 moveto 1   1.5 lineto closepath stroke
newpath 2 3 moveto 2.5 0   lineto closepath stroke
showpage
%%EOF
```

2.
```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 155 95
%%Pages: 1
/s 28.34645 def
%%EndProlog
%%Page: 1 1
s s scale
1 s div setlinewidth
5 s div 5 s div translate
% the triangle with vertices
% (0,0), (5,0), and (2,3)
newpath
0 0 moveto
5 0 lineto
2 3 lineto
closepath
stroke
% the medians
```

```
  newpath 0 0 moveto 3.5 1.5 lineto closepath stroke
  newpath 5 0 moveto 1   1.5 lineto closepath stroke
  newpath 2 3 moveto 2.5 0   lineto closepath stroke
  % the points
  1 setlinecap
  5 s div setlinewidth
  newpath 0 0 moveto 0 0 rlineto stroke
  newpath 5 0 moveto 0 0 rlineto stroke
  newpath 2 3 moveto 0 0 rlineto stroke
  newpath 3.5 1.5 moveto 0 0 rlineto stroke
  newpath 1 1.5 moveto 0 0 rlineto stroke
  newpath 2.5 0 moveto 0 0 rlineto stroke
  newpath 7 3 div 1 moveto 0 0 rlineto stroke
  showpage
  %%EOF
```

EXERCISE 16

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 57 60
%%BeginProlog
/dir {dup cos exch sin} def
/s 28.34645 def
%%EndProlog
s s scale
1 s div setlinewidth
0.9 1.05 translate
newpath
1 0 moveto
 72 dir lineto
144 dir lineto
216 dir lineto
288 dir lineto
closepath
stroke
showpage
%%EOF
```

EXERCISE 17 Apply the nonzero winding number rule to see the difference between the filled region.

EXERCISE 18

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 161 76
%%BeginProlog
/cm {28.34645 mul} def
/box {3 dict begin
% calling structure: x y w box
% purpose: draws an open white box
% with given width w centered around
% the point x y and outlined in black
  /w exch def
  /y exch def
  /x exch def
  gsave
```

```
    0 setgray
    newpath
    x y moveto
    w 2 div w 2 div rmoveto
    w neg 0 rlineto
    0 w neg rlineto
    w  0 rlineto
    closepath
    gsave
      1 setgray
      fill
    grestore
    stroke
  grestore
end } def
%%EndProlog
% draw the line segments in red
10 10 translate
1 0 0 setrgbcolor
newpath
0     0    moveto
1 cm  1 cm lineto
2 cm  0    lineto
3 cm  0    lineto
4 cm  1 cm lineto
5 cm  2 cm lineto
stroke
% draw the open white boxes outlined in black
0 cm 0    0.2 cm box
1 cm 1 cm 0.2 cm box
2 cm 0    0.2 cm box
3 cm 0    0.2 cm box
4 cm 1 cm 0.2 cm box
5 cm 2 cm 0.2 cm box
showpage
%%EOF
```

EXERCISE 19

1. ```
   %!PS-Adobe-3.0 EPSF-3.0
   %%BoundingBox: 0 0 76 76
   38 38 translate
   newpath
   0 0 36 0 360 arc
   30 0 moveto
   0 0 30 360 0 arcn
   gsave
   0.7 setgray
   fill
   grestore
   stroke
   showpage
   %%EOF
   ```
2. ```
   %!PS-Adobe-3.0 EPSF-3.0
   %%BoundingBox: 0 0 40 40
   2 2 translate
   newpath
   ```

```
0 0 36 0 90 arc
0 0 18 90 0 arcn
closepath
gsave
0.75 0.75 1 setrgbcolor
fill
grestore
stroke
showpage
%%EOF
```

3.
```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 86 54
%%BeginProlog
/s 28.34645 def % 1 cm
/lw1 0.7 s div def %
/lw2 1.4 s div def %
/r 3 def % radius (in cm)
%%EndProlog
4 2 translate
s s scale
lw1 setlinewidth
newpath
0 0 moveto
0 0 r 20 0.3 sub 35 0.3 add arc
closepath
stroke
lw2 setlinewidth
newpath
0 0 r 20 35 arc
0 0 r 2 mul 3 div 35 20 arcn
closepath
gsave
1 1 0 setrgbcolor
fill
grestore
stroke
showpage
%%EOF
```

EXERCISE 20

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 119 62
%%BeginProlog
/s 28.34645 def % 1 cm
/r 0.5 def
/lw 1 s div def
%%EndProlog
s s scale
2.1 1.1 translate
lw setlinewidth
newpath
-2 0 moveto
-2 1 0 1 r arct
0 1 lineto
2 1 2 0 r arct
2 0 lineto
```

```
2 -1 0 -1 r arct
0 -1 lineto
-2 -1 -2 0 r arct
closepath
stroke
showpage
%%EOF
```

EXERCISE 21

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 65 65
%%BeginProlog
/dir {dup cos exch sin} def
/scalevec {3 dict begin
/y exch def
/x exch def
/s exch def
s x mul s y mul
end} def
/addvec {4 dict begin
/y2 exch def
/x2 exch def
/y1 exch def
/x1 exch def
x1 x2 add y1 y2 add
end} def
/s 28.34645 def
/c 0.6 def
%%EndProlog
s s scale
1 s div setlinewidth
1.1 1.1 translate
%
0 setgray
newpath
0 0 1 0 360 arc
stroke
newpath
0 -1 moveto
currentpoint c 157 dir scalevec addvec
0 0 c 203 dir scalevec addvec
0 0
curveto
currentpoint c 23 dir scalevec addvec
0 1 c 347 dir scalevec addvec
0 1
curveto
0 0 1 90 -90 arcn
closepath
%0 setgray
fill
newpath 0  0.5 0.1 0 360 arc fill
1 setgray
newpath 0 -0.5 0.1 0 360 arc fill
showpage
%%EOF
```

EXERCISE 24

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 165 105
%%BeginProlog
/angle {exch atan} def
/dir {dup cos exch sin} def
/addvec {    % a b c d on stack
  3 -1 roll  % a c d b
  add        % a c b+d
  3  1 roll  % b+d a c
  add        % b+d a+c
  exch       % a+c b+d
} def
/subvec {% a b c d on stack
  3 -1 roll    % a c d b
  sub neg      % a c b-d
  3  1 roll    % b-d a c
  sub          % b-d a-c
  exch         % a-c b-d
} def
/scalarvec{  % c a b on stack
  3 -1 roll  % a b c
  dup        % a b c c
  3 1 roll   % a c b c
  mul        % a c bc
  3 1 roll   % bc a c
  mul        % bc ac
  exch       % ac bc
} def
/s 28.34645 def
/A {0 0} def /B {5 0} def /C {2 3} def
/trianglepath {newpath A moveto B lineto
  C lineto closepath} def
/closedpoint {1.5 s div 0 360 arc gsave 0 setgray fill grestore stroke} def
/openpoint {1.5 s div0 360 arc gsave 1 setgray fill grestore stroke} def
%%EndProlog
s s scale
1 s div setlinewidth
12 s div 5 s div translate
% draw the triangle with vertices A, B, C
trianglepath stroke
gsave
trianglepath clip
% gamma is the bisector angle at C
/gamma {A C subvec angle 2 div
        B C subvec angle 2 div add} def
newpath
C moveto 10 gamma dir scalarvec rlineto
stroke
grestore
% draw labels
newpath
A closedpoint
B closedpoint
C closedpoint
/Times-Roman findfont 12 s div scalefont setfont
A 0.425 0 subvec moveto (A) show
```

```
B 0.125 0 addvec moveto (B) show
C -0.15 0.125 addvec moveto (C) show
% draw lines to indicate equal angles
newpath
C 0.5 A C subvec angle B C subvec angle arc
stroke
newpath
/phi {A C subvec angle gamma add 2 div} def
C 0.4 phi dir scalarvec addvec moveto
0.2 phi dir scalarvec rlineto
/phi {B C subvec angle gamma add 2 div} def
C 0.4 phi dir scalarvec addvec moveto
0.2 phi dir scalarvec rlineto
stroke
% compute interscection point and draw it.
% we use here the fact that AB is horizontal
% first determine lambda such that
% C + lambda * (gamma dir) =0
% lambda = -y_C/sin(gamma)
newpath
/lambda {C exch pop neg gamma sin div} def
C lambda gamma dir scalarvec addvec closedpoint
showpage
%%EOF
```

EXERCISE 25

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 130 105
%%BeginProlog
/angle {exch atan} def
/dir {dup cos exch sin} def
/addvec {    % a b c d on stack
  3 -1 roll  % a c d b
  add        % a c b+d
  3  1 roll  % b+d a c
  add        % b+d a+c
  exch       % a+c b+d
} def
/subvec {% a b c d on stack
  3 -1 roll    % a c d b
  sub neg      % a c b-d
  3  1 roll    % b-d a c
  sub          % b-d a-c
  exch         % a-c b-d
} def
/scalarvec{  % c a b on stack
  3 -1 roll  % a b c
  dup        % a b c c
  3 1 roll   % a c b c
  mul        % a c bc
  3 1 roll   % bc a c
  mul        % bc ac
  exch       % ac bc
} def
/s 28.34645 def
/A {0 0} def /B {3.75 -0.75} def /C {1.5 2.25} def
```

```
/trianglepath {newpath A moveto B lineto
  C lineto closepath} def
/closedpoint {1.5 s div 0 360 arc gsave 0 setgray fill grestore stroke} def
/openpoint {1.5 s div0 360 arc gsave 1 setgray fill grestore stroke} def
%%EndProlog
s s scale
1 s div setlinewidth
12 s div 5 s div translate
0 0.75 translate
% draw the triangle with vertices A, B, C
trianglepath stroke
gsave
trianglepath clip
newpath
% gamma is the bisector angle at C
/gamma {A C subvec angle 2 div
        B C subvec angle 2 div add} def
C moveto 10 gamma dir scalarvec rlineto
% beta is the bisector angle at C
/beta {A B subvec angle 2 div
        C B subvec angle 2 div add} def
B moveto 10 beta dir scalarvec rlineto
% alpha is the bisector angle at C
/alpha {B A subvec angle 360 sub 2 div
        C A subvec angle 2 div add} def
A moveto 10 alpha dir scalarvec rlineto
stroke
grestore
% draw labels and dots
newpath
A closedpoint
B closedpoint
C closedpoint
/Times-Roman findfont 12 s div scalefont setfont
A 0.425 0 subvec moveto (A) show
B 0.125 0 addvec moveto (B) show
C -0.15 0.15 addvec moveto (C) show
showpage
%%EOF
```

EXERCISE 28

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 130 35
/Helvetica-BoldOblique findfont 40 scalefont setfont
1.5 setlinewidth
newpath
-4 3 moveto
(Yellow) false charpath
clip
gsave 1 1 0 setrgbcolor fill grestore
stroke
showpage
%%EOF
```

EXERCISE 29

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 215 215
%%Pages: 1
/openpoint {2 0 360 arc gsave 1 setgray fill grestore stroke} def
/arrowhead {% stack: s x1 y1, current point: x0 y0
  gsave
    currentpoint   % s x1 y1 x0 y0
    4 2 roll exch  % s x0 y0 y1 x1
    4 -1 roll exch % s y0 y1 x0 x1
    sub 3 1 roll   % s x1-x2 y0 y1
    sub exch       % s y0-y1 x1-x2
    atan rotate    % rotate over arctan((y0-y1)/(x1-x2))
    dup scale      % scale by factor s
    -7 2 rlineto 1 -2 rlineto -1 -2 rlineto
    closepath fill  % fill arrowhead
  grestore
  newpath
} bind def
/f1 {/Times-Italic findfont 15 scalefont setfont} bind def
/f2 {/Times-Roman findfont 12 scalefont setfont} bind def
/f3 {/Times-Roman findfont 15 scalefont setfont} bind def
/f4 {/Times-Roman findfont 32 scalefont setfont} bind def
/fs {/Symbol findfont 18 scalefont setfont} bind def
%%EndProlog
%%Page: 1 1
20 20 translate
newpath % the axes
-15 0 moveto 190 0 lineto 0 -15 moveto 0 190 lineto stroke
gsave % the arrow
  2 setlinewidth
  newpath 0 0 moveto 120 150 lineto stroke
  newpath 124 155 moveto 2 0 0 arrowhead
grestore
.7 setlinewidth
newpath % auxiliary lines
124 0 moveto 124 155 lineto 0 155 lineto stroke
0 0 openpoint % the origin
% the labels
-14 -14 moveto f1 (0) show
120 -12 moveto f1 (a) show
-12 150 moveto f1 (b) show
180 -12 moveto f1 (x) show
-12 180 moveto f1 (y) show
129 155 moveto f3 (\050) show
f1 (a) show f3 (, ) show
f1 (b) show f3 (\051) show
77 70 moveto
gsave 1 2 scale f3 (\050) show grestore
gsave 8 14 rmoveto f1 (a) show grestore
gsave 8 -6 rmoveto f1 (b) show grestore
19 0 rmoveto gsave 1 2 scale f3 (\051) show grestore
showpage
%%EOF
```

EXERCISE 32

In the code below we list the contents of the stack; this explains the result.

```
/display { % stack: r d
dup        % r d d
3 1 roll   % d r d
10 exch    % d r 10 d
exp        % d r 10^d
mul        % d r*10^d
round      % d round(r*10^d)
10         % d round(r*10^d) 10
3 -1 roll  % round(r*10^d) 10 d
exp        % round(r*10^d) 10^d
div        % round(r*10^d)/10^d
} def
```

EXERCISE 33

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 173 31
%%BeginProlog
/u 14.173225 def % unit is 0.5 cm
/lw {1 u div} def % default linewidth
/n 10 def
/circle { % arguments: graylevel x y
  0.25 0 360 arc setgray fill } def
%%EndProlog
%%Page: 1 1
u dup scale
lw setlinewidth
1.1 1.1 translate
newpath % draw border
-1 -1 moveto
n 2 add 0 rlineto
0 2 rlineto
-2 n sub 0 rlineto
closepath
stroke
newpath
0 1 n {
  /i exch def
  % draw filled circle
  i n div i 0 circle
} for
showpage
%%EOF
```

EXERCISE 34

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 500 225
%%BeginProlog
/u 16 def % 16 point per unit
/lw {1 u div} def % default linewidth
/Times-Roman findfont 8 u div scalefont setfont
/display {dup 3 1 roll 10 exch exp mul round
```

```
          10 3 -1 roll exp div} def
/str {10 string} def
/n 10 def
/dp 2 def % display precision
%%EndProlog
u dup scale
lw setlinewidth
1.8 1.8 translate
% left diagram
0 1 n {
  /i exch def
  i 0.1 add -0.75 moveto
  i n div dp display str cvs show % draw horizontal label
  -1 i 0.25 add moveto
  i n div dp display str cvs show % draw vertical label
  0 1 n {
    /j exch def
    gsave  % draw filled rectangle
      i n div j n div add 3 div dup dup setrgbcolor
      newpath i j 1 1 rectfill
      0 setgray
      newpath i j 1 1 rectstroke
    grestore
    } for
} for
n 2 div -1.5 moveto (r) show
-1.5 n 2 div moveto (g) show
n 2 div 1.5 sub n 1.5 add moveto (normal gray shading) show
%
n 3 add 0 translate
% right diagram
0 1 n {
  /i exch def
  i 0.1 add -0.75 moveto
  i n div dp display str cvs show % draw horizontal label
  -1 i 0.25 add moveto
  i n div dp display str cvs show % draw vertical label
  0 1 n {
    /j exch def
    gsave  % draw filled rectangle
      0.3 i n div mul 0.59 j n div mul add dup dup setrgbcolor
      newpath i j 1 1 rectfill
      0 setgray
      newpath i j 1 1 rectstroke
    grestore
    } for
} for
n 2 div -1.5 moveto (r) show
-1.5 n 2 div moveto (g) show
n 2 div 1 sub n 1.5 add moveto (tv gray shading) show
showpage
%%EOF
```

EXERCISE 35

```
AFPL Ghostscript 8.00 (2002-11-21)
Copyright (C) 2002 artofcode LLC, Benicia, CA. All rights reserved.
```

```
This software comes with NO WARRANTY: see the file PUBLIC for details.
GS>/i -1 def
GS>/j  1 def
GS>i j gt {i}{j} ifelse
GS<1>pstack
1
GS<1>
```

EXERCISE 36

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 102 102
%%BeginProlog
/triangle { 5 dict begin
  /z exch def
  /y exch def
  /x exch def
  x y z add le
  y z x add le and
  z x y add le and {
     /cosalpha {x 2 exp z 2 exp add y 2 exp sub 2 div x div z div} def
     /sinalpha {1 cosalpha 2 exp sub sqrt} def
     newpath
     x 0 moveto
     0 0 lineto
     z cosalpha mul z sinalpha mul lineto
     closepath
     stroke
  } if
  end } def
%%EndProlog
%%Page: 1 1
50 2 translate
30 60 40 triangle
10 20 60 triangle % no triangle possible
0 0 -1 triangle % no triangle possible
showpage
%%EOF
```

EXERCISE 38

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 120 112
%%BeginProlog
/u 54 def % unit: 0.5 inch
/lw 1 u div def
/point {2.5 lw mul 0 360 arc
  gsave 0 setgray fill grestore stroke
} bind def
/makepolygon {2 dict begin
  /vertices exch def
  /n vertices length def
  n 1 gt {
    vertices 0 get aload pop moveto % move to first vertex
    vertices 1 n 1 sub getinterval % get subarray
    { aload pop lineto} forall % make lines
```

```
  } if
end} bind def
/drawvertices {
  {aload pop point newpath} forall % make vertices
} bind def
/vertices [ [0 1] [234 cos 234 sin] [18 cos 18 sin]
  [162 cos 162 sin] [306 cos 306 sin] ] def
%%EndProlog
u dup scale 1.1 0.95 translate
lw setlinewidth 2 setlinejoin
newpath vertices makepolygon closepath stroke
vertices drawvertices
showpage
%%EOF
```

EXERCISE 39

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 120 120
%%BeginProlog
/u 28.34645 def % unit: 1cm
%%EndProlog
u dup scale
1 u div setlinewidth
2.1 2.1 translate
newpath
-2 0 moveto 4 0 rlineto
0 -2 moveto 0 4 rlineto
-2 1 2 {/i exch def
  i -0.2 moveto 0 0.4 rlineto
  -0.2 i moveto 0.4 0 rlineto
} for
-2 0.2 2 {/i exch def
  i -0.1 moveto 0 0.2 rlineto
  -0.1 i moveto 0.2 0 rlineto
} for
stroke
showpage
%%EOF
```

EXERCISE 40

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 170 140
%%BeginProlog
/u 2.834645 def    % unit: 1 mm
/lw1 {1 u div} def   % normal line width
/lw2 {0.5 u div} def % line width for 5 mm lines
/lw3 {0.2 u div} def % line width for 1 mm lines
/makepaper {5 dict begin % stack: nx ny
  /ny exch def % number of horizontal boxes of size 1 cm
  /nx exch def % number of vertical boxes of size 1 cm
  /w 10 nx mul def % graph paper width
  /h 10 ny mul def % graph paper height
  % draw vertical lines
  0 1 10 nx mul {
```

```
    /i exch def
    i 0 moveto
    i 10 mod 0 eq
      {gsave lw1 setlinewidth 0 h rlineto stroke grestore}
      {i 5 mod 0 eq
         {gsave lw2 setlinewidth 0 h rlineto stroke grestore}
         {gsave lw3 setlinewidth 0 h rlineto stroke grestore
      } ifelse
    } ifelse
  } for
  % draw horizontal lines
  0 1 10 ny mul {
    /i exch def
    0 i moveto
    i 10 mod 0 eq
      {gsave lw1 setlinewidth w 0 rlineto stroke grestore}
      {i 5 mod 0 eq
         {gsave lw2 setlinewidth w 0 rlineto stroke grestore}
         {gsave lw3 setlinewidth w 0 rlineto stroke grestore
      } ifelse
    } ifelse
  } for
end} def
%%EndProlog
u dup scale
lw1 setlinewidth
5 5 translate
1 0 0 0 setcmykcolor
newpath
5 4 makepaper
showpage
%%EOF
```

EXERCISE 41

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 330 535
%%BeginProlog
/u 28.34645 def      % unit: 1 cm
/lw1 {0.8 u div} def % normal line width
/lw2 {0.4 u div} def % line width for 5 mm lines
/lw3 {0.2 u div} def % line width for 5 mm lines
/sy 9 def % size of one vertical cycle
/logcoord {log sy mul} bind def
/hf1 {/Helvetica findfont 8 u div scalefont setfont} bind def
/hf2 {/Helvetica findfont 5 u div scalefont setfont} bind def
/makelogpaper {5 dict begin % stack: nx ny
  /ny exch def % number of vertical cycles
  /nx exch def % number of horizontal units
  /h {sy ny mul} def
  /w {nx} def
  % draw vertical lines
  0 1 10 nx mul {
    /i exch def
    i 10 div 0 moveto
    i 10 mod 0 eq
      {gsave lw1 setlinewidth 0 -0.4 rmoveto 0 h 0.4 add rlineto stroke grestore}
```

```
      {i 5 mod 0 eq
         {gsave lw2 setlinewidth 0 -0.2 rmoveto 0 h 0.2 add rlineto stroke grestore}
         {gsave lw3 setlinewidth 0 h rlineto stroke grestore
      } ifelse
    } ifelse
  } for
  % draw horizontal lines
  0 1 ny 1 sub {
    /i exch def
    % draw lines between 1 and 5
    0 1 40 {
      /j exch def
      0 i sy mul moveto
      j 10 mod 0 eq
        {gsave lw1 setlinewidth -0.4 1 j 10 div add logcoord rmoveto
               w 0.4 add 0 rlineto stroke grestore}
        {j 5 mod 0 eq
           {gsave lw2 setlinewidth -0.2 1 j 10 div add logcoord rmoveto
                  w 0.2 add 0 rlineto stroke grestore}
           {gsave lw3 setlinewidth 0 1 j 10 div add logcoord rmoveto
                w 0 rlineto stroke grestore
        } ifelse
      } ifelse
    } for
    % draw lines between 5 and 10
    1 1 25 {
      /j exch def
      0 i sy mul moveto
      j 5 mod 0 eq
        {gsave lw1 setlinewidth -0.4 5 j 5 div add logcoord rmoveto
               w 0.4 add 0 rlineto stroke grestore}
        {gsave lw3 setlinewidth 0 5 j 5 div add logcoord rmoveto
               w 0 rlineto stroke grestore
      } ifelse
    } for
  } for
  % draw vertical labels
  0 1 ny 1 sub {/i exch def
    -0.6 i sy mul 0.1 sub moveto
    i 0 eq
      {hf1 (1) show}
      {-0.2 0 rmoveto hf1 (10) show
       0 0.2 rmoveto hf2 i 1 string cvs show
    } ifelse
    2 1 9 {
      /j exch def
      -0.6 i sy mul 0.1 sub moveto
      0 j logcoord rmoveto
      hf1 j 1 string cvs show
    } for
  } for
  -0.6 h 0.1 sub moveto
  -0.2 0 rmoveto hf1 (10) show
  0 0.2 rmoveto hf2 ny 1 string cvs show
end} def
%%EndProlog
u dup scale
```

```
lw1 setlinewidth
1 0.5 translate
1 0 0 0 setcmykcolor
newpath
10 2 makelogpaper
showpage
%%EOF
```

EXERCISE 42

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 570 565
%%BeginProlog
/u 28.34645 def      % unit: 1 cm
/lw1 {0.8 u div} def % normal line width
/lw2 {0.4 u div} def % line width for 5 mm lines
/lw3 {0.2 u div} def % line width for 5 mm lines
/s 9 def % size of one vertical cycle
/logcoord {log s mul} bind def
/hf1 {/Helvetica findfont 8 u div scalefont setfont} bind def
/hf2 {/Helvetica findfont 5 u div scalefont setfont} bind def
/makelogpaper {5 dict begin % stack: nx ny
  /ny exch def % number of vertical cycles
  /nx exch def % number of horizontal units
  /h {s ny mul} def
  /w {s nx mul} def
  % draw horizontal lines
  0 1 ny 1 sub {
    /i exch def
    % draw lines between 1 and 2
    0 1 10 {
      /j exch def
      0 i s mul moveto
      j 10 mod 0 eq
        {gsave lw1 setlinewidth -0.4 1 j 10 div add logcoord rmoveto
               w 0.8 add 0 rlineto stroke grestore}
        {j 5 mod 0 eq
           {gsave lw2 setlinewidth -0.2 1 j 10 div add logcoord rmoveto
                  w 0.4 add 0 rlineto stroke grestore}
           {gsave lw3 setlinewidth 0 1 j 10 div add logcoord rmoveto
                 w 0 rlineto stroke grestore
        } ifelse
      } ifelse
    } for
    % draw lines between 2 and 4
    1 1 10 {
      /j exch def
      0 i s mul moveto
      j 5 mod 0 eq
        {gsave lw1 setlinewidth -0.4 2 j 5 div add logcoord rmoveto
               w 0.8 add 0 rlineto stroke grestore}
        {gsave lw3 setlinewidth 0 2 j 5 div add logcoord rmoveto
               w 0 rlineto stroke grestore
      } ifelse
    } for
    % draw lines between 4 and 10
    1 1 12 {
```

```
    /j exch def
    0 i s mul moveto
    j 2 mod 0 eq
      {gsave lw1 setlinewidth -0.4 4 j 2 div add logcoord rmoveto
             w 0.8 add 0 rlineto stroke grestore}
      {gsave lw3 setlinewidth -0.2 4 j 2 div add logcoord rmoveto
             w 0.4 add 0 rlineto stroke grestore
    } ifelse
  } for
} for
% draw vertical lines
0 1 nx 1 sub {
  /i exch def
  % draw lines between 1 and 2
  0 1 10 {
    /j exch def
    i s mul 0 moveto
    j 10 mod 0 eq
      {gsave lw1 setlinewidth 1 j 10 div add logcoord -0.4 rmoveto
             0 h 0.8 add rlineto stroke grestore}
      {j 5 mod 0 eq
         {gsave lw2 setlinewidth 1 j 10 div add logcoord -0.2 rmoveto
                0 h 0.4 add rlineto stroke grestore}
         {gsave lw3 setlinewidth 1 j 10 div add logcoord 0 rmoveto
              0 h rlineto stroke grestore
      } ifelse
    } ifelse
  } for
  % draw lines between 2 and 4
  1 1 10 {
    /j exch def
    i s mul 0 moveto
    j 5 mod 0 eq
      {gsave lw1 setlinewidth 2 j 5 div add logcoord -0.4 rmoveto
             0 h 0.8 add rlineto stroke grestore}
      {gsave lw3 setlinewidth 2 j 5 div add logcoord 0 rmoveto
             0 h rlineto stroke grestore
    } ifelse
  } for
  % draw lines between 4 and 10
  1 1 12 {
    /j exch def
    i s mul 0 moveto
    j 2 mod 0 eq
      {gsave lw1 setlinewidth 4 j 2 div add logcoord -0.4 rmoveto
             0 h 0.8 add  rlineto stroke grestore}
      {gsave lw3 setlinewidth 4 j 2 div add logcoord -0.2 rmoveto
             0 h 0.4 add rlineto stroke grestore
    } ifelse
  } for
} for
% draw horizontal labels
% down
0 1 nx 1 sub {/i exch def
  i s mul 0.08 sub -0.8 moveto
  i 0 eq
    {hf1 (1) show}
```

```
    {-0.1 0 rmoveto hf1 (10) show
     0 0.2 rmoveto hf2 i 1 string cvs show
  } ifelse
  2 1 9 {
    /j exch def
    i s mul 0.08 sub -0.8 moveto
    j logcoord 0 rmoveto
    hf1 j 1 string cvs show
  } for
% up
  i s mul 0.08 sub h 0.5 add moveto
  i 0 eq
    {hf1 (1) show}
    {-0.1 0 rmoveto hf1 (10) show
     0 0.2 rmoveto hf2 i 1 string cvs show
  } ifelse
  2 1 9 {
    /j exch def
    i s mul 0.08 sub h 0.5 add moveto
    j logcoord 0 rmoveto
    hf1 j 1 string cvs show
  } for
} for
h 0.08 sub -0.8 moveto
-0.1 0 rmoveto hf1 (10) show
0 0.2 rmoveto hf2 ny 1 string cvs show
h 0.08 sub h 0.5 add moveto
-0.1 0 rmoveto hf1 (10) show
0 0.2 rmoveto hf2 ny 1 string cvs show
% draw vertical labels
% left
0 1 ny 1 sub {/i exch def
  -0.6 i s mul 0.1 sub moveto
  i 0 eq
    {hf1 (1) show}
    {-0.2 0 rmoveto hf1 (10) show
     0 0.2 rmoveto hf2 i 1 string cvs show
  } ifelse
  2 1 9 {
    /j exch def
    -0.6 i s mul 0.1 sub moveto
    0 j logcoord rmoveto
    hf1 j 1 string cvs show
  } for
% right
  w 0.5 add i s mul 0.1 sub moveto
  i 0 eq
    {hf1 (1) show}
    {hf1 (10) show
     0 0.2 rmoveto hf2 i 1 string cvs show
  } ifelse
  2 1 9 {
    /j exch def
    w 0.5 add i s mul 0.1 sub moveto
    0 j logcoord rmoveto
    hf1 j 1 string cvs show
  } for
```

```
  } for
  -0.6 h 0.1 sub moveto
  -0.2 0 rmoveto hf1 (10) show
  0 0.2 rmoveto hf2 ny 1 string cvs show
  w 0.5 add h 0.1 sub moveto
  hf1 (10) show
  0 0.2 rmoveto hf2 ny 1 string cvs show

end} def
%%EndProlog
u dup scale
lw1 setlinewidth
1 1 translate
1 0 0 0 setcmykcolor
newpath
2 2 makelogpaper
showpage
%%EOF
```

EXERCISE 43

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 100 130
%%BeginProlog
/u 28.34645 def    % unit: 1 cm
/concat { % stack: (a) (b) -> (ab)
  exch dup length
  2 index length add string
  dup dup 4 2 roll copy length
  4 -1 roll putinterval
} bind def
/xoff 3 def /yoff 2 def % absolute value of offset of label
/dotlabel-rt {% stack: s x y
  moveto currentpoint
  gsave newpath 2 u div 0 360 arc fill grestore
  xoff u div yoff neg u div rmoveto show
} def
/dotlabel-lft {% stack: s x y
  moveto currentpoint
  gsave newpath 2 u div 0 360 arc fill grestore
  dup stringwidth pop xoff neg u div exch sub yoff neg u div rmoveto show
} def
/K {3 dict begin % stack: n
  /n exch def %
  newpath
  1 1 n {/i exch def
    1 1 n {/j exch def
      0 i 1 sub moveto n 2 div j 1 sub lineto
      0 i 1 sub moveto (a) i 2 string cvs concat currentpoint dotlabel-lft
      n 2 div i 1 sub moveto (b) i 2 string cvs concat currentpoint dotlabel-rt
    } for
  } for
  stroke
end} def

%%EndProlog
u dup scale
```

```
0.5 0.25 translate
1 u div setlinewidth
2 setlinejoin
/Times-Roman findfont 10 u div scalefont setfont
5 K
showpage
%%EOF
```

EXERCISE 44

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 140 100
%%BeginProlog
/u 28.343452 def % unit: 1 cm
/lw1 {1 u div} def
/lw2 {1.5 u div} def
/fr {/Helvetica findfont 10 u div scalefont setfont} bind def
/fi {/Helvetica-Italic findfont 10 u div scalefont setfont} bind def
/fs {/Symbol findfont 10 u div scalefont setfont} bind def
/f { % the function f(x)
  1 dict begin
  /x exch def
  x sqrt % sqrt(x)
end } bind def
/xmin 0 def /xmax 4 def /ymin 0 def /ymax 2 def
/txmin xmin ceiling def  /txmax xmax floor def % ticks
/tymin ymin ceiling def  /tymax ymax floor def
/dx 0.01 def % stepsize
%%EndProlog
u dup scale
0.75 1 translate
lw1 setlinewidth 2 setlinecap
% draw axes
newpath
xmin 0 moveto xmax 0 lineto
0 ymin moveto 0 ymax lineto
stroke
% draw ticks and axes labels
newpath
txmin 1 txmax {/i exch def
  i 0 moveto 0 0.1 rlineto
  -0.1 -0.5 rmoveto fr i 1 string cvs show
} for
tymin 1 tymax {/i exch def
  0 i moveto 0.1 0 rlineto
  -0.4 -0.1 rmoveto fr i 1 string cvs show
} for
% draw text
txmax txmin sub 2 div 0.1 sub -0.8 moveto fi (x) show
-0.6 tymax tymin sub 2 div 0.1 sub moveto fi (y) show
2.5 1.2 moveto fi (y) show fr ( = ) show fs (\326) show fi (x) show
3.08 1.21 moveto fs (\140) show
stroke
% draw graph
0 0 1 setrgbcolor
lw2 setlinewidth 0 setlinecap
/x xmin def
```

```
newpath
x x f moveto
{/x x dx add def x xmax 0.0001 add gt {exit} if
 x x f lineto } loop
stroke
showpage
%%EOF
```

EXERCISE 45

1.
```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 380 78
%%BeginProlog
/u 28.343452 def % unit: 1 cm
/pi 3.14159265 def
/lw1 {1 u div} def
/lw2 {1.5 u div} def
/fr {/Helvetica findfont 10 u div scalefont setfont} bind def
/fi {/Helvetica-Italic findfont 10 u div scalefont setfont} bind def
/fs {/Symbol findfont 10 u div scalefont setfont} bind def
/f {180 mul pi div sin} bind def
/nmin -2 def /nmax 2 def
/xmin nmin pi mul def /xmax nmax pi mul def
/ymin -1.2 def /ymax 1.2 def
/n 75 def % number of segments
/dx {xmax xmin sub n div} bind def %stepsize
%%EndProlog
u dup scale
-0.5 xmin add neg -0.2 ymin add neg translate
lw1 setlinewidth
2 setlinecap
% draw axes
newpath
xmin 0 moveto xmax 0 lineto
0 ymin moveto 0 ymax lineto
stroke
% draw ticks
newpath
3 nmin mul 1 3 nmax mul {
  /i exch def
  i 0 ne {
  i 3 div pi mul 0 moveto 0 0.1 rlineto} if
} for
-1 1 1 {/i exch def
  i 0 ne {0 i moveto 0.1 0 rlineto} if
} for
% draw text (manually)
-0.4 -1.1 moveto fr (-1) show
-0.3 0.9 moveto fr (1) show
-6.7 -0.4 moveto fs (-2p) show
-3.4 -0.4 moveto fs (-p) show
3.05 -0.4 moveto fs (p) show
6.2 -0.4 moveto fs (2p) show
3 0.5 moveto fi (y) show fr ( = sin ) show fi (x) show
stroke
% draw sine graph
0 0 1 setrgbcolor
```

```
lw2 setlinewidth 0 setlinecap
/x xmin def
newpath
x x f moveto
n {
  /x x dx add def
  x x f lineto
} repeat
stroke
showpage
%%EOF
```

2.
```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 380 78
%%BeginProlog
/u 28.343452 def % unit: 1 cm
/pi 3.14159265 def
/lw1 {1 u div} def
/lw2 {1.5 u div} def
/fr {/Helvetica findfont 10 u div scalefont setfont} bind def
/fi {/Helvetica-Italic findfont 10 u div scalefont setfont} bind def
/fs {/Symbol findfont 10 u div scalefont setfont} bind def
/f {180 mul pi div sin} bind def
/f' {180 mul pi div cos} bind def
/nmin -2 def /nmax 2 def
/xmin nmin pi mul def /xmax nmax pi mul def
/ymin -1.2 def /ymax 1.2 def
/n 15 def % number of segments
/dx {xmax xmin sub n div} bind def %stepsize
%%EndProlog
u dup scale
-0.5 xmin add neg -0.2 ymin add neg translate
lw1 setlinewidth
2 setlinecap
% draw axes
newpath
xmin 0 moveto xmax 0 lineto
0 ymin moveto 0 ymax lineto
stroke
% draw ticks
newpath
3 nmin mul 1 3 nmax mul {
  /i exch def
  i 0 ne {
  i 3 div pi mul 0 moveto 0 0.1 rlineto} if
} for
-1 1 1 {/i exch def
  i 0 ne {0 i moveto 0.1 0 rlineto} if
} for
% draw text (manually)
-0.4 -1.1 moveto fr (-1) show
-0.3 0.9 moveto fr (1) show
-6.7 -0.4 moveto fs (-2p) show
-3.4 -0.4 moveto fs (-p) show
3.05 -0.4 moveto fs (p) show
6.2 -0.4 moveto fs (2p) show
3 0.5 moveto fi (y) show fr ( = sin ) show fi (x) show
stroke
```

```
  % draw sine graph
  0 0 1 setrgbcolor
  lw2 setlinewidth 0 setlinecap
  /x xmin def
  newpath
  x x f moveto
  n {
    x dx 3 div add
    x f dx 3 div x f' mul add % P1
    /x x dx add def
    x dx 3 div sub
    x f dx 3 div x f' mul sub % P2
    x x f % P3
    curveto
  } repeat
  stroke
  showpage
  %%EOF
```

EXERCISE 46

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 540 270
%%BeginProlog
/u 28.343452 def % unit: 1 cm
/pi 3.14159265 def
/lw1 {1 u div} def
/lw2 {1.5 u div} def
/fr {/Helvetica findfont 10 u div scalefont setfont} bind def
/fi {/Helvetica-Italic findfont 10 u div scalefont setfont} bind def
/fs {/Symbol findfont 10 u div scalefont setfont} bind def
/rad {180 pi div mul} def
/f {dup rad sin exch rad cos div} bind def % tangens
/xmin -7 def /xmax 7 def
/ymin -4.2 def /ymax 4.2 def
/txmin xmin pi div ceiling cvi def  % used for ticks
/txmax xmax pi div floor cvi def
/tymin ymin ceiling cvi def
/tymax ymax floor cvi def
/n 2500 def % number of segments
/dx {xmax xmin sub n div} bind def %stepsize
%%EndProlog
u dup scale
-0.5 xmin add neg -0.2 ymin add neg translate
lw1 setlinewidth
2 setlinecap
% draw axes
newpath
xmin 0 moveto xmax 0 lineto
0 ymin moveto 0 ymax lineto
stroke
% draw ticks and axes labels
newpath
txmin 1 txmax {
  /i exch def
  i 0 gt {
    i pi mul 0 moveto 0 0.1 rlineto
```

```
    i 1 eq {
      -0.1 -0.5 rmoveto fs (p) show} {
      -0.1 -0.5 rmoveto fs i 2 string cvs show fs (p) show
    } ifelse
  } if
  i 0 lt {
    i pi mul 0 moveto 0 0.1 rlineto
    i -1 eq {
      -0.1 -0.5 rmoveto fs (-p) show} {
      -0.1 -0.5 rmoveto fs i 2 string cvs show fs (p) show
    } ifelse
  } if
  i 0 eq { 0.1 -0.4 moveto fs (0) show } if
} for
tymin 1 tymax {/i exch def
  i 0 lt {
    0 i moveto 0.1 0 rlineto
   -0.55 -0.1 rmoveto i 2 string cvs show } if
  i 0 gt {
    0 i moveto 0.1 0 rlineto
   -0.37 -0.1 rmoveto i 2 string cvs show } if
} for
% draw text (manually)
2.9 .9 moveto fr (tan) show .1 0 rmoveto fi (x) show
stroke
% draw asymptotic lines
0 setgray [.3 .2] 0 setdash
lw1 setlinewidth
newpath
txmin 1 txmax {
  /i exch def
  i 0 lt {
    2 i mul 1 add pi mul 2 div ymin moveto
    0 ymax ymin sub rlineto
  } if
  i 0 gt {
    2 i mul 1 sub pi mul 2 div ymin moveto
    0 ymax ymin sub rlineto
  } if
} for
stroke
% draw graph of tangens
0 0 1 setrgbcolor [] 0 setdash
lw2 setlinewidth 0 setlinecap
/x xmin def
newpath
x x f moveto
n {
  /x x dx add def
  x f ymin gt x f ymax lt and
    {x x f lineto }
    { x x f moveto}
  ifelse
} repeat
stroke
showpage
%%EOF
```

EXERCISE 47

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 230 175
%%BeginProlog
/u 28.34645 def % unit: 1 cm
/e 2.71828 def  % base of natural logarithm
/lw1 {1 u div} def
/lw2 {1.5 u div} def
/fr {/Times-Roman findfont 10 u div scalefont setfont} bind def
/fi {/Helvetica-Italic findfont 10 u div scalefont setfont} bind def
/fis {/Helvetica-Italic findfont 7 u div scalefont setfont} bind def
/str 10 string def
/f { % the function f(x)
  1 dict begin
  /x exch def
  e x exp 1 x add div % f(x) = exp(x)/(1+x)
end } bind def
/coord {% coordinates in logplot
  % stack: x y
  log 2.5 mul    % x 2.5*log10(y)
} bind def
/n 100 def
/xmax 6 def
/ny 2 def
/ymax 10 ny exp def % always choose a power of 10
%%EndProlog
u dup scale
% translate origin
1 0.6 translate
% draw axes
lw1 setlinewidth
newpath
0 1 coord moveto xmax 0.5 add 1 coord lineto
0 1 coord moveto 0 ymax coord 0.5 add lineto
xmax 0.3 add -0.4 moveto fi (x) show
-0.3 ymax coord 0.3 add moveto fi (y) show
stroke
% draw ticks
newpath
0 1 xmax floor cvi  {/i exch def
  i 1 coord moveto 0 0.1 rlineto
  -0.1 -0.5 rmoveto fr i str cvs show
} for
1 1 ny {/i exch def
  1 1 10 {/j exch def
    /yj {10 i 1 sub exp j mul cvi } def % yj=j*10^(i-1)
    0 yj coord moveto 0.1 0 rlineto
    j 5 mod 0 eq { % j a 5-fold
       /lbl yj str cvs def
       -0.2 lbl stringwidth pop sub -0.1 rmoveto
       fr lbl show
    } if
  } for
} for
-0.3 -0.1 moveto fr (1) show
% draw text
0.5 50 coord moveto fr (graph of ) show
```

```
gsave 0 -0.3 rmoveto fr ( 1 + ) show fi (x) show grestore
gsave 0.02 0.1 rmoveto fr (_____) show grestore
0.3 0.2 rmoveto fi (e) show
0.04 0.15 rmoveto fis (x) show
stroke
% draw graph
lw2 setlinewidth
0 0 1 setrgbcolor 2 setlinecap
/x 0 def
/dx xmax n div def
newpath
x x f coord moveto %
n {
  /x x dx add def
  x x f coord
  lineto
} repeat
stroke
showpage
%%EOF
```

EXERCISE 48

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 190 130
%%BeginProlog
/u 28.34645 def % unit: 1 cm
/lw1 {1 u div} def
/lw2 {1.5 u div} def
/fs {/Symbol findfont 10 u div scalefont setfont} bind def
/fi {/Helvetica-Italic findfont 10 u div scalefont setfont} bind def
/fis {/Helvetica-Italic findfont 7 u div scalefont setfont} bind def
/str 10 string def
/f { % the function f(x)
  1 dict begin
  /x exch def
  1 x div
end } bind def
/n 100 def
/xmin 0.25 def /xmax 6 def
/ymax 1 xmin div def
/dx xmax xmin sub n div def
%%EndProlog
u dup scale
lw1 setlinewidth
% translate origin
0.5 0.5 translate
% draw graph
lw2 setlinewidth
2 setlinecap
/x xmin def
newpath
x x f moveto %
5 {
  /x x dx add def
  x x f
  lineto
```

```
} repeat
stroke
newpath
/xleft x def
x x f moveto
50 {
  /x x dx add def
  x x f
  lineto
} repeat
gsave
  0 x f neg rlineto
  xleft x sub 0 rlineto
  closepath
  gsave 0.8 setgray fill grestore
  lw1 setlinewidth 0 setgray
  stroke
grestore
n 60 sub {
  /x x dx add def
  x x f
  lineto
} repeat
stroke
% draw axes
lw1 setlinewidth 0 setgray
newpath
0 0 moveto xmax 0 lineto
0 0 moveto 0 ymax lineto
xmax 0.3 sub -0.4 moveto fi (x) show
-0.3 ymax 0.3 sub moveto fi (y) show
stroke
showpage
%%EOF
```

EXERCISE 49

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 270 150
%%BeginProlog
/u 28.34645 def % unit: 1 cm
/lw1 {1 u div} def
/lw2 {1.5 u div} def
/f { % the function f(x)
  1 dict begin
  /x exch def
  4 x dup mul sub % 4 - x^2
end } bind def
/n 100 def
/xmin -2.1 def /xmax 2.1 def
/ymin -0.5 def /ymax 4.5 def
/dx xmax xmin sub n div def
/nr 12 def % number of rectangles
/x0 -2 def /x1 2 def % left and right bound
/inc x1 x0 sub nr div def
/min { 2 copy gt { exch } if pop } bind def
/max { 2 copy lt { exch } if pop } bind def
```

```
%%EndProlog
u dup scale
lw1 setlinewidth
% translate origin for left picture
2.4 0.6 translate
% draw axes
lw1 setlinewidth
newpath
xmin 0 moveto xmax 0 lineto
0 ymin moveto 0 ymax lineto
stroke
% draw reactangles
lw1 setlinewidth 2 setlinejoin
0 setgray
newpath
/x x0 def
/fxprev x f def
x 0 moveto
nr {
  /x x inc add def
  inc 0 rlineto
  0 x f fxprev max rlineto
  inc neg 0 rlineto
  closepath
  gsave 1 1 0 setrgbcolor fill grestore
  x 0 moveto
  /fxprev x f def
} repeat
stroke
% draw graph
lw2 setlinewidth 2 setlinecap
/x xmin def
newpath
x x f moveto %
n {
  /x x dx add def
  x x f
  lineto
} repeat
stroke
%
% translate origin for right picture
4.8 0 translate
% draw axes
lw1 setlinewidth
newpath
xmin 0 moveto xmax 0 lineto
0 ymin moveto 0 ymax lineto
stroke
% draw reactangles
lw1 setlinewidth 2 setlinejoin
0 setgray
newpath
/x x0 def
/fxprev x f def
x 0 moveto
nr {
```

```
  /x x inc add def
  inc 0 rlineto
  0 x f fxprev min rlineto
  inc neg 0 rlineto
  closepath
  gsave 1 1 0 setrgbcolor fill grestore
  x 0 moveto
  /fxprev x f def
} repeat
stroke
% draw graph
lw2 setlinewidth 2 setlinecap
/x xmin def
newpath
x x f moveto %
n {
  /x x dx add def
  x x f
  lineto
} repeat
stroke
showpage
%%EOF
```

EXERCISE 53

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 65 75
%%BeginProlog
/u 28.34645 def % unit: 1 cm
/ellipse {7 dict begin
  /endangle exch def
  /startangle exch def
  /yradius exch def
  /xradius exch def
  /yC exch def
  /xC exch def
  /savematrix matrix currentmatrix def % save current transformation matrix
  xC yC translate                      % translate to center of ellipse
  xradius yradius scale                % scale by radius values
  0 0 1 startangle endangle arc        % add arc to path
  savematrix setmatrix                 % restore the transformation matrix
end
} def
%%EndProlog
u dup scale
1 u div setlinewidth
2.7 2.3 translate
gsave
  -2 0 translate
  0 0 0.5 0.25 0 360 ellipse
  stroke
grestore
gsave
  60 rotate
  -2 0 translate
  0 0 0.5 0.25 0 360 ellipse
```

```
  stroke
grestore
-1.5 0 moveto
0 0 1.5 180 240 arc
-2.5 0 moveto
0 0 2.5 180 240 arc
stroke
showpage
%%EOF
```

EXERCISE 54

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 143 124
%%BeginProlog
/u 20 def % 20 points per unit
/r 4 def
/phi 3 def
/s 1 phi sin 3 sqrt mul phi cos add div def
/triangle {
  newpath
  0 r moveto
  r 210 cos mul r 210 sin mul lineto
  r 330 cos mul r 330 sin mul lineto
  closepath fill } def
/b 1 def         % amount of blue
/white false def % start with blue
%%EndProlog
72 42 translate
u dup scale
1 u div setlinewidth
triangle
180 phi idiv {
   white {0 0 1 sethsbcolor}{b 1 1 sethsbcolor} ifelse
   triangle
   /white white not def % change color
   /r s r mul def % shrink triangle
   /b s b mul def % change color
   phi rotate } repeat
showpage
%%EOF
```

EXERCISE 55

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 100 110
%%BeginProlog
/u 0.25 def
/lw 1 u div def
%%EndProlog
60 56 translate
u dup scale
lw setlinewidth
newpath
3 {
  125 170 moveto
```

```
  125 -170 lineto
  85 -191 lineto
  85 100 lineto
  -88 0 lineto
  120 rotate
} repeat
stroke
showpage
%%EOF
```

EXERCISE 56

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 247 154
%%BeginProlog
/phi 5 sqrt 1 add 2 div def % golden ratio
/s 150 def % size of original square
/square {
  newpath 0.6 setlinewidth 0 setgray
  0 0 moveto s 0 rlineto 0 s rlineto s neg 0 rlineto
  closepath stroke
  newpath 1.5 setlinewidth
  0.647 0.165 0.165 setrgbcolor
  s 0 s 180 90 arcn stroke
} def
%%EndProlog
2 2 translate
20 {
  square
  s s translate
  -90 rotate
  /s s phi div def
} repeat
showpage
%%EOF
```

EXERCISE 57

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 173 150
%%BeginProlog
/n 4 def % how many iterations
/u 147 def % 147 points per unit (about 3*sqrt(3) cm)
/lw 1 u div def
/side {
    dup     % store current depth
    0 gt {  % test if depth is already 0
      1 sub % reduce depth by 1
      1 3 div dup scale % scale to 1/3
      /lw 3 lw mul def % adjust linewidth
      lw setlinewidth
      %%%%% draw one side as components
      side
      60 rotate side
      -120 rotate side
      60 rotate side %
```

```
      3 dup scale % back to original scale
      /lw lw 3 div def % reset linewidth
      lw setlinewidth
      1 add % restore prev depth to stack
    }
    {
      newpath
      %%%%% draw line from 0,0 to 0,1 in current coordinate system
      0 0 moveto 0 1 lineto stroke
      0 1 translate
    }
    ifelse
} bind def

/Koch { % draw initial triangle
    side
    -120 rotate side
    -120 rotate side
    pop % clear stack
} bind def
%%EndProlog
u dup scale
lw setlinewidth
1 setlinecap
0.3 0.01 translate
n Koch
showpage
showpage
%%EOF
```

EXERCISE 58

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 0 0 66 66
%%BeginProlog
/n 3 def % how many iterations
/s 9 def
/lw 1 s div def
% reflect about line from (0,0) to (1,-1)
/R {90 rotate -1 1 scale} bind def
/invR { -1 1 scale -90 rotate} bind def
% reflect about line from (0,0) to (1,1)
/M {-1 1 scale 90 rotate} def
/invM { -90 rotate -1 1 scale } def
/H {
  dup 0 gt {
    1 sub
    M H invM 1 0 rlineto
    H 0 1 rlineto
    H -1 0 rlineto
    R H invR
    1 add
  } if
} bind def
%%EndProlog
2 2 translate
s dup scale
```

```
lw setlinewidth
newpath
2 setlinecap
0 0 moveto 0 0 rlineto
n H
stroke
showpage
%%EOF
```