```
/* Same with frac.rr */
import("glib3.rr")$
extern CX$
extern CY$
def lineto(T,R) {
  extern CX;
  extern CY;
  X=deval(CX + R*cos(T));
  Y=deval(CY + R*sin(T));
  glib_line(CX,CY,X,Y);
  CX=X; CY=Y;
}
def moveto(X,Y) {
  extern CX;
  extern CY;
  CX=X; CY=Y;
}
def ccurve2(T,R,N) {
  print([T,R,N]);
  if (N<2) {
     lineto(T,R); return(0);
  }
  ccurve2(T+@pi/4,R/2^(1/2),N-1);
  ccurve2(T-@pi/4,R/2^(1/2),N-1);
}
def main(N) {
  glib_window(-10,-10,15,15);
  glib_clear();
  moveto(0,0);
  ccurve2(0,10,N);
 glib_flush();
}
end$

/*
glib_clear();
glib_window(-1,-1,10,10);
moveto(0,0); glib_flush();
lineto(@pi/4,5); glib_flush();
lineto(0,3); glib_flush();
main(2);
main(3);
main(4);
main(5);
main(10);
*/
```



```
/*
  cc ccurve-r.c glib4.c -lX11 -lm
*/
#include <stdio.h>
#include <math.h>
#include "glib4.h"
double CX;
double CY;
void mylineto(double T,double R) {
  extern double CX;
  extern double CY;
  double X,Y;
  X=CX + R*cos(T);
  Y=CY + R*sin(T);
  glib_color(NULL,0xff0000);
  glib_line(CX,CY,X,Y);
  CX=X; CY=Y;
}
void mymoveto(double X,double Y) {
  extern double CX;
  extern double CY;
  CX=X; CY=Y;
}
void ccurve2(double T,double R,int N) {
  if (N<2) {
     mylineto(T,R); return;
  }
  ccurve2(T+M_PI/4,R/sqrt(2),N-1);
  ccurve2(T-M_PI/4,R/sqrt(2),N-1);
}
int main() {
  glib_open();
  glib_window(-10,-10,20,15);
  glib_loop();
}
void glib_draw() {
  int n;
  n=8;
  /* n=20;  */
  glib_clear();
  mymoveto(0,0);
  ccurve2(0,10,n);
  glib_flush();
}


/*
glib_clear();
glib_window(-1,-1,10,10);
moveto(0,0); glib_flush();
lineto(@pi/4,5); glib_flush();
lineto(0,3); glib_flush();
main(2);
main(3);
main(4);
main(5);
main(10);
*/
```