**8.A) Write a program to implements Cohen-Sutherland Clipping.**

**Code:**

#include<stdio.h>

#include<stdlib.h>

#include<graphics.h>

#define MAX 20

enum{TOP =0x1,BOTTOM=0x2,RIGHT=0x4,LEFT=0x8};

enum{FALSE,TRUE};

typedef unsigned int outcode;

outcode compute\_outcode(int x,int y, int xmin,int ymin, int xmax,int ymax){

outcode oc=0;

if(y>ymax)

oc=TOP;

if(y<ymin)

oc=BOTTOM;

if(x>xmax)

oc=RIGHT;

else if(x<xmin)

oc=LEFT;

return oc;

}

void cohen\_sutherland(double x1,double y1,double x2,double y2, double xmin,double ymin,double xmax,double ymax){

int accept;

int done;

outcode outcode1,outcode2;

accept=FALSE;

done=FALSE;

outcode1=compute\_outcode(x1,y1,xmin,ymin,xmax,ymax);

outcode2=compute\_outcode(x2,y2,xmin,ymin,xmax,ymax);

do{

if(outcode1==0&&outcode2==0){

accept=TRUE;

done=TRUE;

}

else if(outcode1 & outcode2){

done=TRUE;

}

else{

double x,y;

int outcode\_ex=outcode1?outcode1:outcode2;

if(outcode\_ex&TOP) {

x=x1+(x2-x1)\*(ymax-y1)/(y2-y1);

y=ymax;

}

else if(outcode\_ex&BOTTOM) {

x=x1+(x2-x1)\*(ymin-y1)/(y2-y1);

y=ymin;

}

else if(outcode\_ex&RIGHT) {

y=y1+(y2-y1)\*(xmax-x1)/(x2-x1);

x=xmax;

}

else {

y=y1+(y2-y1)\*(xmin-x1)/(x2-x1);

x=xmin;

}

if(outcode\_ex==outcode1){

x1=x;

y1=y;

outcode1=compute\_outcode(x1,y1,xmin,ymin,xmax,ymax);

}

else{

x2=x;

y2=y;

outcode2=compute\_outcode(x2,y2,xmin,ymin,xmax,ymax);

}

}

}while(done==FALSE);

if(accept==TRUE){

line(x1,y1,x2,y2);

}

}

void main()

{

int n;

int i,j;

int ln[MAX][4];

int clip[4];

int gd=DETECT,gm;

clrscr();

printf("Enter the number of lines to be clipped:\n");

scanf("%d",&n);

printf("Enter the x-and y-coordinates of the line end-points:\n");

for(i=0;i<n;i++)

{

for(j=0;j<4;j++)

{

scanf("%d",&ln[i][j]);

}

}

printf("Enter the x-and y-coordinates of the line end-points left-top and right:\n");

printf("bottom corners\n of the clip window:\n");

for(i=0;i<4;i++)

{

scanf("%d",&clip[i]);

}

initgraph(&gd,&gm,"C://TC//BGI");

rectangle(clip[0],clip[1],clip[2],clip[3]);

for(i=0;i<n;i++)

line(ln[i][0],ln[i][1],ln[i][2],ln[i][3]);

getch();

cleardevice();

rectangle(clip[0],clip[1],clip[2],clip[3]);

for(i=0;i<n;i++)

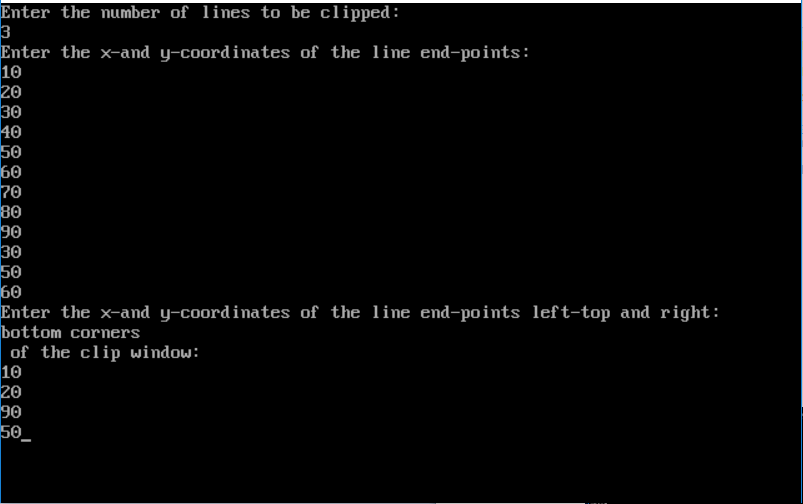
{

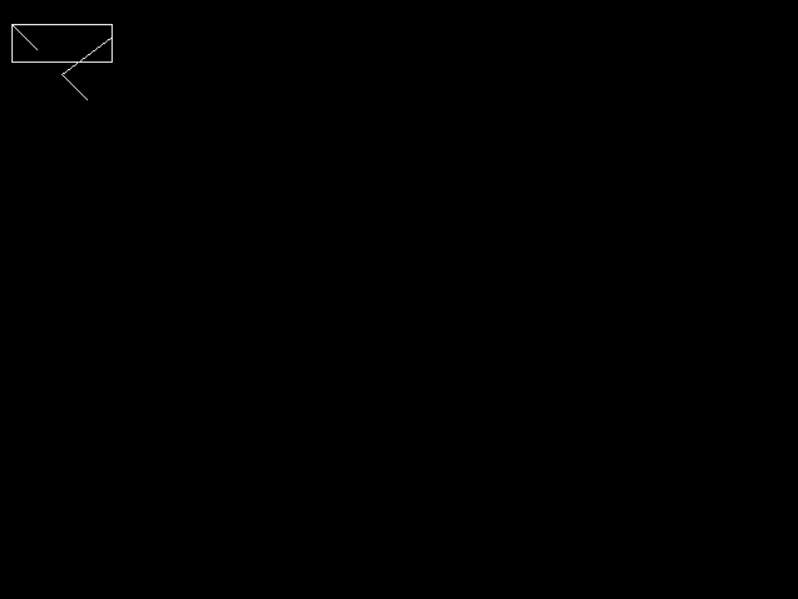
cohen\_sutherland(ln[i][0],ln[i][1],ln[i][2],ln[i][3],clip[0],clip[1],clip[2],clip[3]);

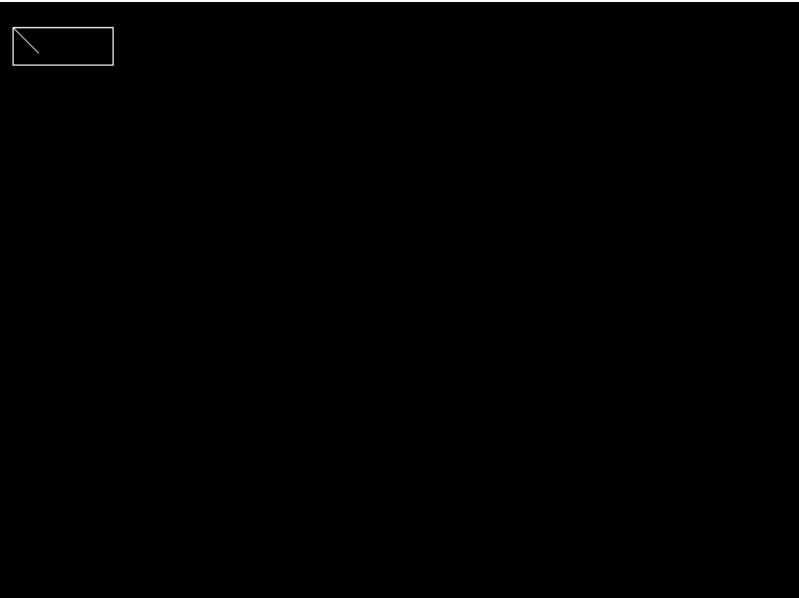
getch(); }

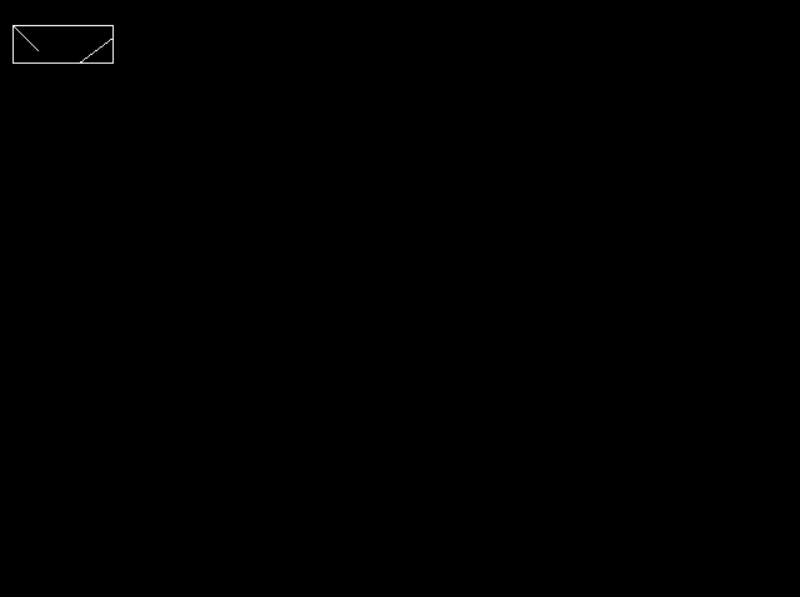
closegraph(); }

**Output:**









**8.B) Write a program to implement Liang-Barsky Line Clipping Algorithm.**

**Code:**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

void main()

{

int i,gd=DETECT,gm;

int x1,y1,x2,y2,xmin,ymin,ymax,xmax,xx1,xx2,yy1,yy2,dx,dy;

float t1,t2,p[4],q[4],temp;

clrscr();

printf("Enter line coordinates x1, y1:");

scanf("%d %d",&x1, &x1);

printf("Enter line coordinates x2, y2:");

scanf("%d %d",&x2, &x2);

xmin=100;

ymin=100;

xmax=250;

ymax=250;

initgraph(&gd,&gm,"C:\\TC\\BGI");

rectangle(xmin ymin,xmax,ymax);

dx=x2-x1;

dy=y2-y1;

p[0]=-dx;

p[1]=dx;

p[2]=-dy;

p[3]=dy;

q[0]=x1-xmin;

q[1]=xmax-x1;

q[2]=y1-ymin;

q[3]=ymax-1;

for(i=0;i<4;i++)

{

if(p[i]==0)

{

printf("Line is parallel to one of the clipping boundary");

if(q[i]>=0)

{

if(i<2)

{

if(y1<ymin){

y1=ymin;

}

if(y2>ymax){

y2=ymax;

}

line(x1,y1,x2,y2);

}

if(i>1)

{

if(x1<xmin){

x1=xmin;

}

if(x2>xmax){

x2=xmax;

}

line(x1,y1,x2,y2);

}

}

}

}

t1=0;

t2=1;

for(i=0;i<4;i++)

{

temp=q[i]/p[i];

if(p[i]<0){

if(t1<=temp)

t1=temp;

}

else{

if(t2>temp)

t2=temp;

}

}

if(t1<t2){

xx1=x1+t1\*p[1];

xx2=x1+t2\*p[1];

yy1=y1+t1\*p[3];

yy2=y1+t2\*p[3];

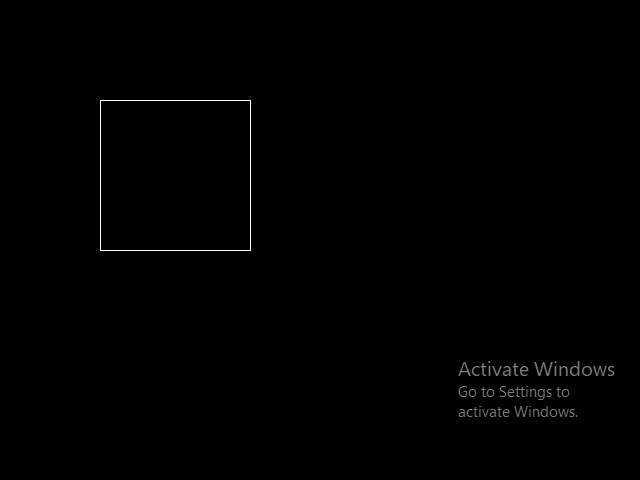
line(xx1,yy1,xx2,yy2);

} delay(1000);

getch();

closegraph();}

**Output:**



**9.A)Write a program to fill a circle using Flood Fillm Algorithm.**

**code:**

#include<stdio.h>

#include<graphics.h>

void floodFill(int x, int y, int oldcolor, int newcolor)

{

if(getpixel(x,y)==oldcolor)

{

putpixel(x,y,newcolor);

floodFill(x+1,y,oldcolor,newcolor);

floodFill(x,y+1,oldcolor,newcolor);

floodFill(x-1,y,oldcolor,newcolor);

floodFill(x,y-1,oldcolor,newcolor);

}

}

void main()

{

int gm,gd=DETECT, radius;

int x,y;

clrscr();

printf("enter x and y positions for circle\n");

scanf("%d%d",&x,&y);

printf("enter radius of circle\n");

scanf("%d",&radius);

initgraph(&gd,&gm,"C:\\TC\\BGI");

circle(x,y,radius);

floodFill(x,y,0,15);

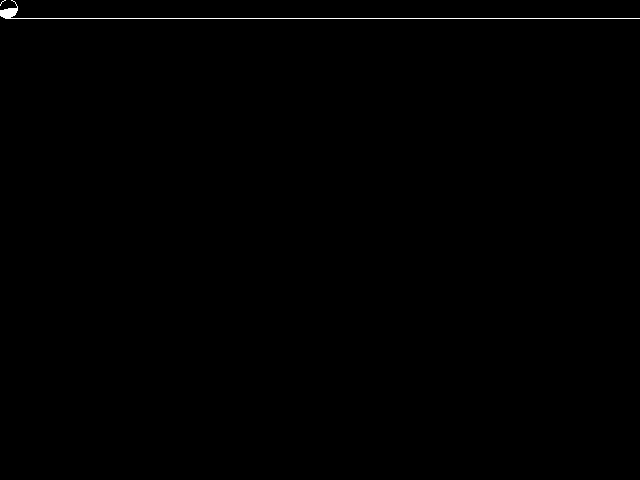
delay(5000);

closegraph();

getch();

}

**Output:**



**9.B)Write a program to fill using Boundary Fill Algorithm.**

**code:**

#include<stdio.h>

#include<graphics.h>

void boundaryfill(int x, int y,int f\_color,int b\_color)

{

if(getpixel(x,y)!=b\_color&& getpixel(x,y)!=f\_color)

{

putpixel(x,y,f\_color);

boundaryfill(x+1,y,f\_color,b\_color);

boundaryfill(x,y+1,f\_color,b\_color);

boundaryfill(x-1,y,f\_color,b\_color);

boundaryfill(x,y-1,f\_color,b\_color);

}

}

void main(){

int gm,gd=DETECT,radius;

int x,y;

clrscr();

printf("enter x and y positions for circle\n");

scanf("%d%d", &x,&y);

printf("enter radius of the circle\n");

scanf("%d",&radius);

initgraph(&gd,&gm,"C:\\TC\\BGI");

circle(x,y,radius);

boundaryfill(x,y,4,15);

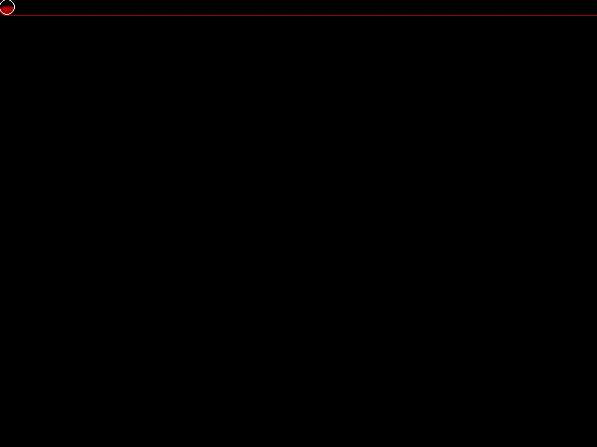
delay(5000);

closegraph();

getch();

}

**Output:**



**10.A) Develop a simple text screen sever using graphics Function.**

**Code:**

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main(){

int gd=DETECT,gm,x=600,i;

initgraph(&gd,&gm,"C:\\TURBO3\\BGI");

//setting limit of movement of text screen saver here we are setting it as 200 means it will move upto 250 times,if we increase this it will run in output screen upto that limit,thus we wont be able to come out of output screen which indicates its still under execution not in infinite loop kindly note.

for (x=0;x<250;x++)

{

x%=250;

setcolor(random(16));

circle(random(635),random(70),50);

circle(random(635),random(70),50);

circle(random(635),random(70),50);

circle(random(635),random(70),50);

circle(random(635),random(70),50);

clearviewport();

settextstyle(1,0,5);

setcolor(RED);

outtextxy(50,415-2\*x,"\*WORLD\*");

setcolor(GREEN);

outtextxy(200,415-2\*x,"\*of\*");

setcolor(YELLOW);

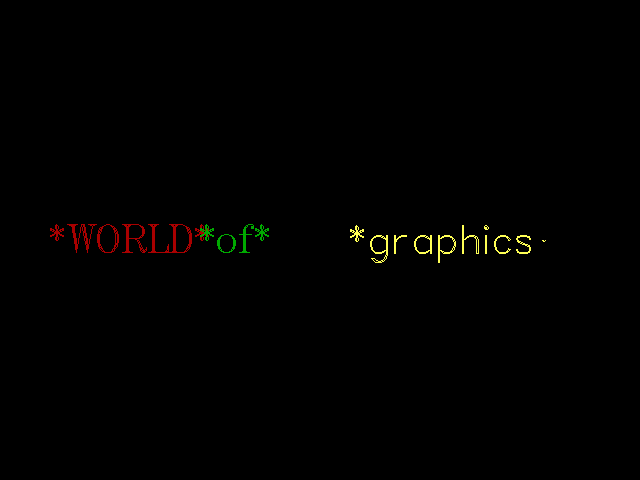
settextstyle(3,0,5);

outtextxy(350,415-2\*x,"\*graphics\*");

}

getch(); }

**Output:**



**10.B) Perform Smiling face animation using graphics function.**

**Code:**

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

void main()

{int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

//for head

circle(200,200,30);

//for left eye

circle(190,190,5);

arc(190,190,50,130,10);

//for right eye

circle(210,190,5);

arc(210,190,50,130,10);

//for smiley lips

arc(200,210,180,360,10);

line(187,210,193,210);

line(207,210,213,210);

//for nose

line(198,195,195,200);

line(202,195,205,200);

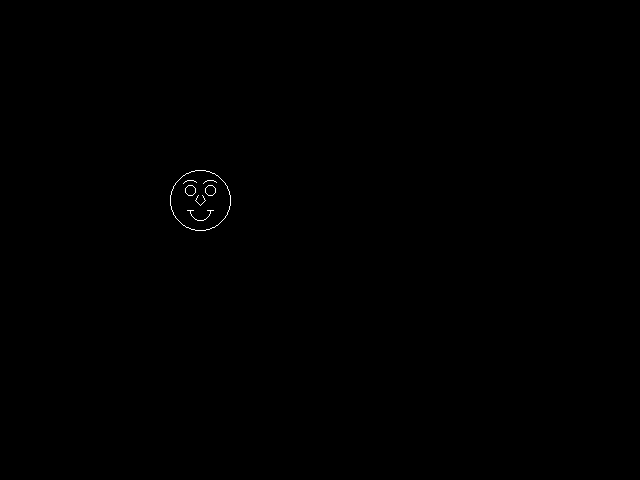
line(195,200,200,205);

line(205,200,200,205);

getch();

closegraph();

}**Output:**



**10.C) Draw the moving car on the screen.**

**Code:**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

void main()

{

int gd=DETECT,gm;

int i,maxx, midy;

initgraph(&gd,&gm,"C:\\TC\\BGI");

maxx=getmaxx();

midy=getmaxx()/2;

for(i=0;i<maxx-150;i=i+5)

{

cleardevice();

setcolor(WHITE);

line(0,midy+37,maxx,midy+37);

setcolor(YELLOW);

setfillstyle(SOLID\_FILL,RED);

line(i,midy+23,i,midy);

line(i,midy,40+i,midy-20);

line(40+i,midy-20,80+i,midy-20);

line(80+i,midy-20,100+i,midy);

line(100+i,midy,120+i,midy);

line(120+i,midy,120+i,midy+23);

line(0+i,midy+23,18+i,midy+23);

arc(30+i,midy+23,0,180,12);

line(42+i,midy+23,78+i,midy+23);

arc(90+i,midy+23,0,180,12);

line(120+i,midy+23,120+i,midy+23);

line(28+i,midy,43+i,midy-15);

line(43+i,midy-15,57+i,midy-15);

line(57+i,midy-15,57+i,midy);

line(57+i,midy,28+i,midy);

line(62+i,midy-15,77+i,midy-15);

line(77+i,midy-15,92+i,midy);

line(92+i,midy,62+i,midy);

line(62+i,midy,62+i,midy-15);

floodfill(5+i,midy+22,YELLOW);

setcolor(BLUE);

setfillstyle(SOLID\_FILL,DARKGRAY);

circle(30+i,midy+25,9);

circle(90+i,midy+25,9);

floodfill(30+i,midy+25,BLUE);

floodfill(90+i,midy+25,BLUE);

delay(100);

}

getch();

closegraph();

}

**Output:**

