**PROGRAM 1**

**Name:-** *Kaustubh S Kabra*

**Class:-***Second Year Engineering Comp-1*

**Roll No:-** *20*

**Write a c++ program for drawing graphics primitive and color it.**

#include<graphics.h>

#include<conio.h>

void main()

{

clrscr();

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

// Line

line(25,50,100,250);

//Rectange

setfillstyle(SOLID\_FILL,GREEN);

rectangle(125,50,275,200);

floodfill(126,55,WHITE);

//circle

setfillstyle(SOLID\_FILL,RED);

circle(300,350,100);

floodfill(301,355,WHITE);

//Arc

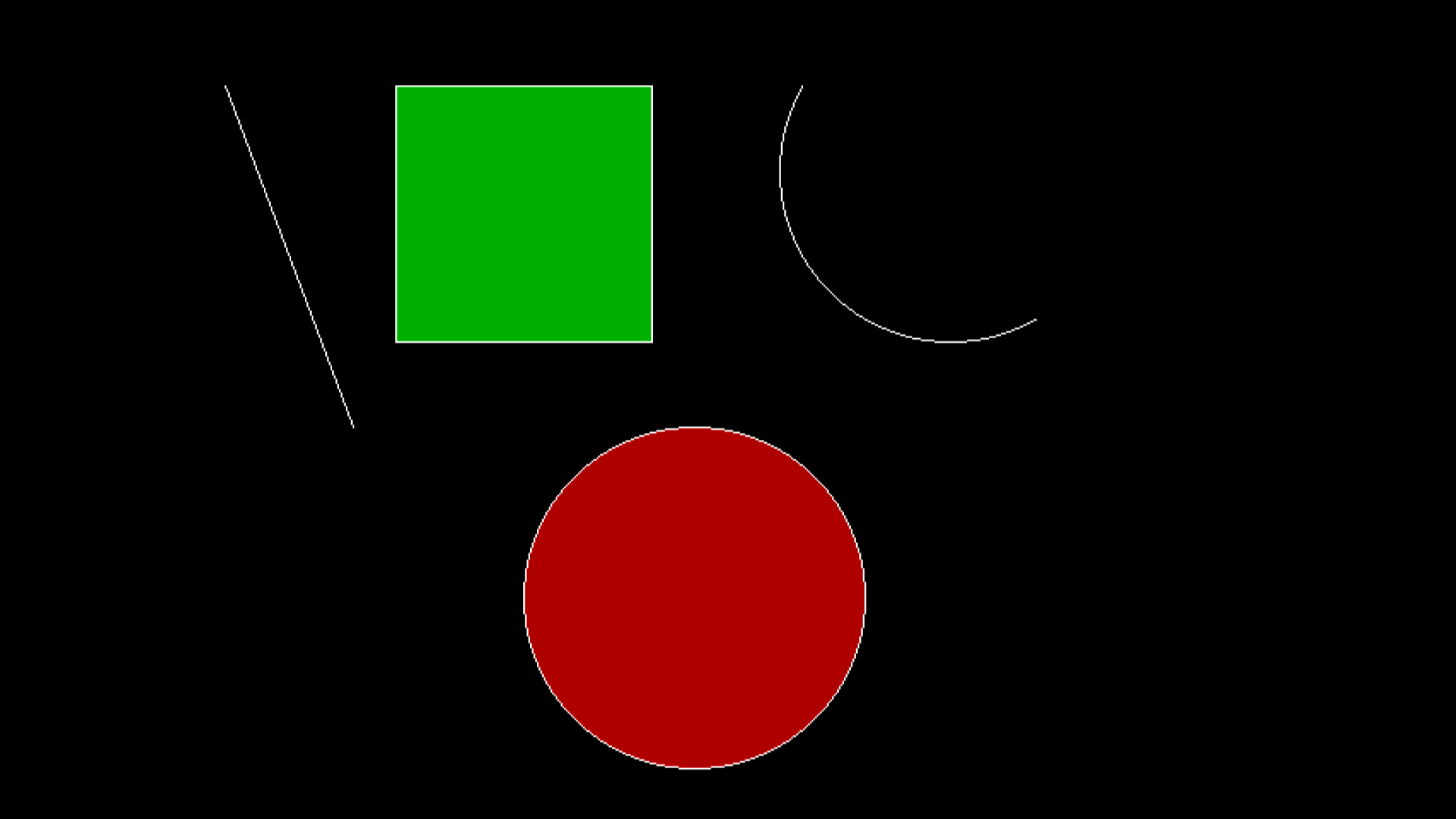
arc(450,100,150,300,100);

getch();

closegraph();

}

**Output:-**



**PROGRAM 2**

**Name:-** *Kaustubh S Kabra*

**Class:-** *Second Year Engineering Comp-1*

**Roll No:-** *20*

**Write a c++ program to divide the screen into four regions and draw a circle,rectangle,arc and ellipse.**

#include<graphics.h>

#include<conio.h>

#include<iostream.h>

void main()

{

clrscr();

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

line(0,getmaxy()/2,getmaxx(),getmaxy()/2);

line(getmaxx()/2,0,getmaxx()/2,getmaxy());

//ellipse in quadrant 1

setfillstyle(SOLID\_FILL,BLUE);

ellipse(450,175,0,360,100,50);

floodfill(450,175,WHITE);

//circle in quadrant 2

setfillstyle(SOLID\_FILL,RED);

circle(100,100,100);

floodfill(100,100,WHITE);

//arc in quadrant 3

arc(450,350,60,240,100);

//rectange in quadrant 4

setfillstyle(SOLID\_FILL,GREEN);

rectangle(50,300,300,450);

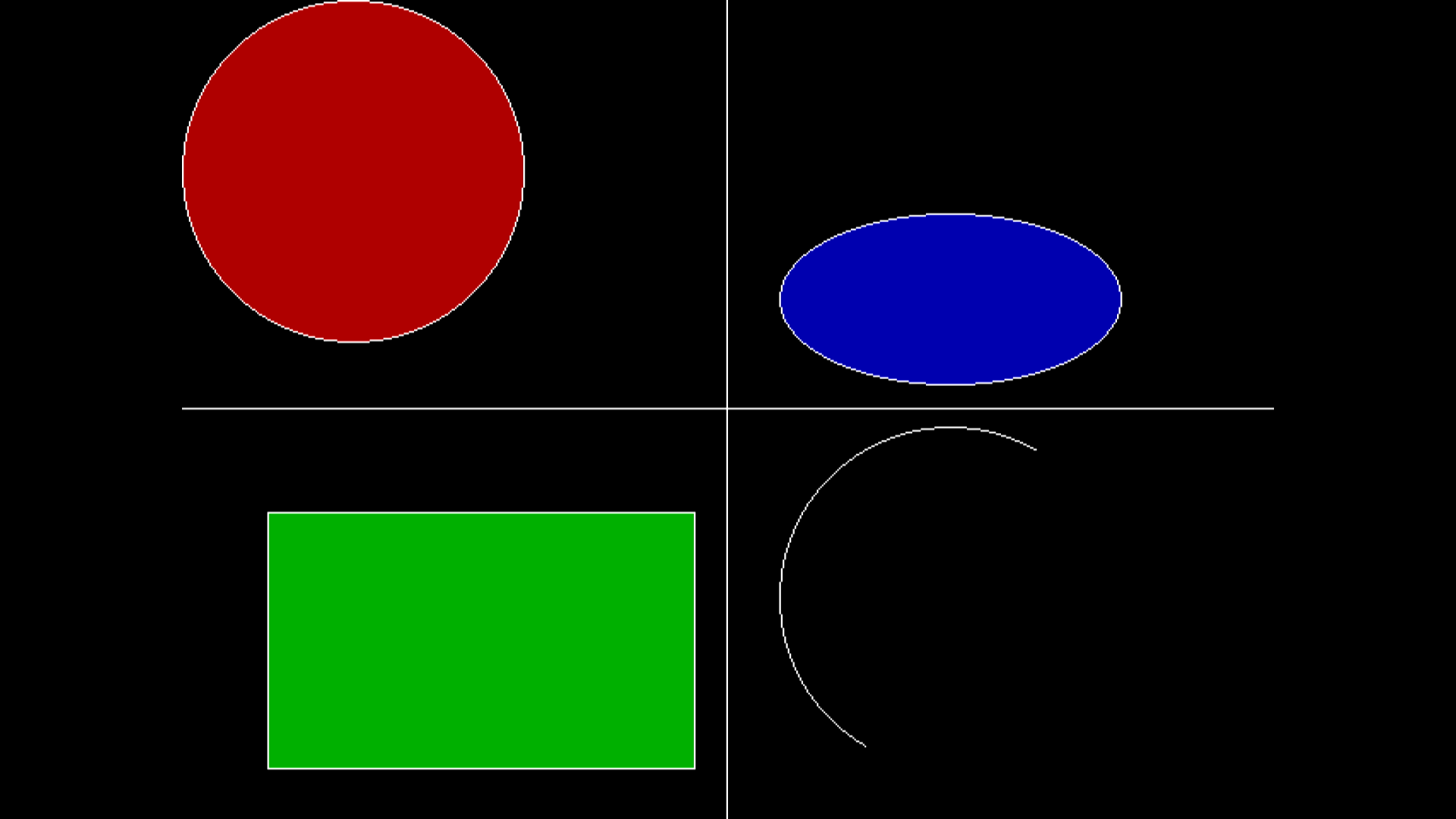
floodfill(51,305,WHITE);

getch();

closegraph();

}

**Output:-**



**PROGRAM 3**

**Name:-** *Kaustubh S Kabra*

**Class:-** *Second Year Engineering Comp-1*

**Roll No:-** *20*

**Write a c++ program for drawing a simple object.**

#include <graphics.h>

#include <conio.h>

void main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

setcolor(YELLOW);

setfillstyle(SOLID\_FILL, YELLOW);

circle(300, 200, 100);

floodfill(300, 200, YELLOW);

setcolor(RED);

setfillstyle(SOLID\_FILL,BLACK);

fillellipse(325, 175, 10, 15);

fillellipse(275, 175, 10, 15);

ellipse(300, 250, 205, 335, 20, 10);

ellipse(300, 250, 205, 335, 20, 11);

ellipse(300, 250, 205, 335, 20, 12);

ellipse(300, 250, 205, 335, 20, 13);

ellipse(300, 250, 205, 335, 20, 14);

ellipse(300, 250, 205, 335, 20, 15);

ellipse(300, 250, 205, 335, 20, 16);

ellipse(300, 250, 205, 335, 20, 17);

ellipse(300, 250, 205, 335, 20, 18);

ellipse(300, 250, 205, 335, 20, 19);

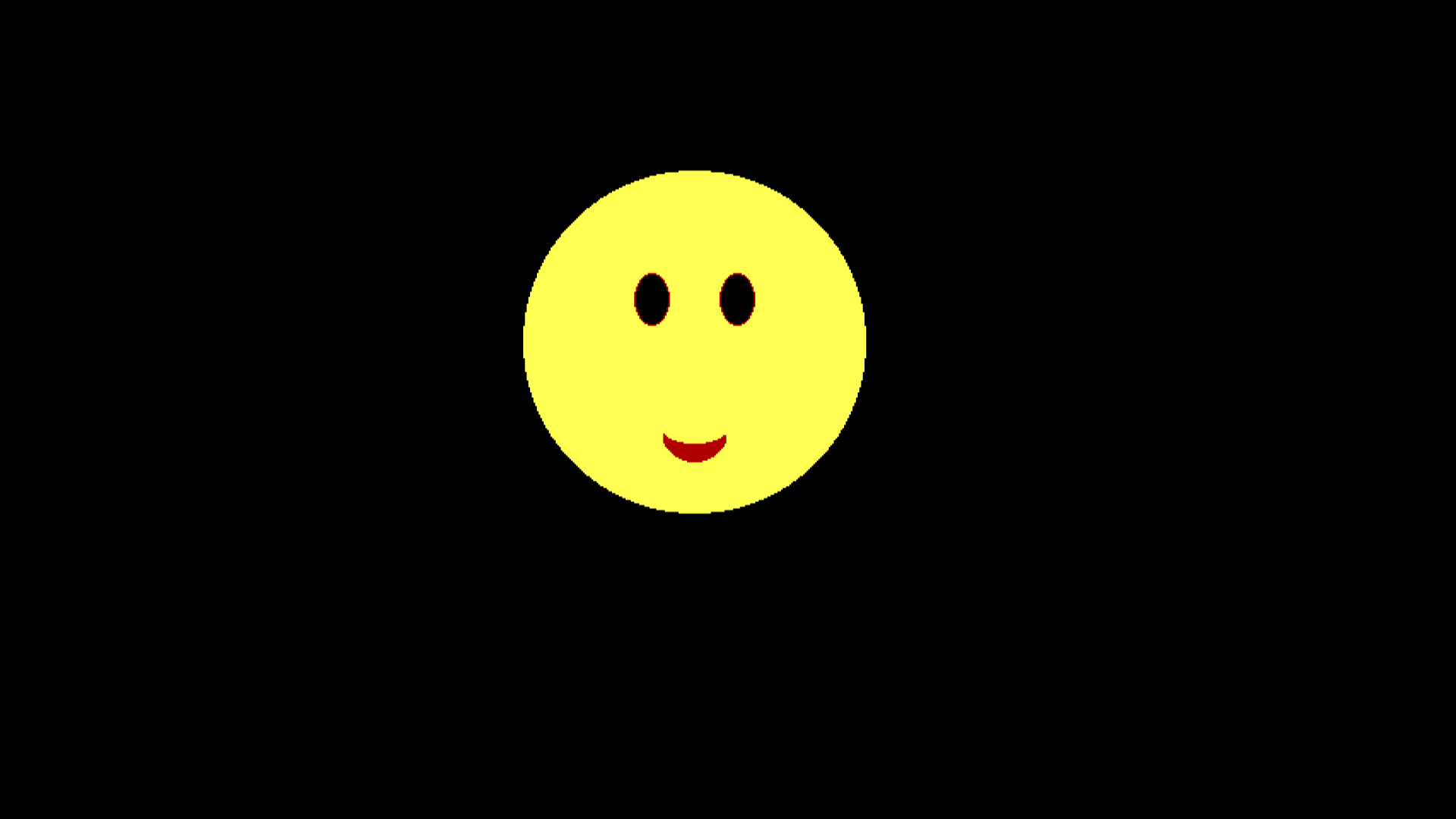
ellipse(300, 250, 205, 335, 20, 20);

getch();

closegraph();

}

**OUTPUT:-**



**PROGRAM 4**

**Name:-** *Kaustubh S Kabra*

**Class:-** *Second Year Engineering Comp-1*

**Roll No:-** *20*

**Write a c++ program for drawing a line using DDA and Bresenhams Line Drawing Algorithm**

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<iostream.h>

int sign(int x)

{

if(x>0)

return 1;

else if(x<0)

return -1;

else

return 0;

}

void dda(int x1,int y1,int x2,int y2,int col)

{

float x,y,l,i;

float dx,dy;

if(x1==x2 && y1==y2)

{

putpixel(x1,x2,col);

}

else

{

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx>=dy)

l=dx;

else

l=dy;

dx=(x2-x1)/l;

dy=(y2-y1)/l;

x=x1+0.5\*sign(dx);

y=y1+0.5\*sign(dy);

i=1;

while(i<l)

{

putpixel(x,y,col);

x=x+dx;

y=y+dy;

i++;

}

}

}

void bla(int x1,int y1,int x2,int y2,int col)

{

int dx,dy,x,y,e,i;

if(x1==x2 && y1==y2)

putpixel(x1,y1,col);

else

{

dx=abs(x2-x1);

dy=abs(y2-y1);

x=x1;

y=y1;

putpixel(x,y,col);

e=2\*dy-dx;

i=1;

while(i<=dx)

{

while(e>=0)

{

y=y+1;

e=e-2\*dx;

}

x=x+1;

e=e+2\*dy;

putpixel(x,y,col);

i=i+1;

}

}

}

void main()

{

clrscr();

int x1,x2,y1,y2,col,c;

cout<<"1)DDA Algorithm"<<endl;

cout<<"2)Bresenham's Algorithm"<<endl;

cout<<"Enter your choice"<<endl;

cin>>c;

switch(c)

{

case 1:

{

cout<<"Enter start cordinates"<<endl;

cin>>x1>>y1;

cout<<"Enter end cordinates"<<endl;

cin>>x2>>y2;

cout<<"Enter colour(1-15)"<<endl;

cin>>col;

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

dda(x1,y1,x2,y2,col);

break;

}

case 2:

{

cout<<"Enter start cordinates"<<endl;

cin>>x1>>y1;

cout<<"Enter end cordinates"<<endl;

cin>>x2>>y2;

cout<<"Enter colour(1-15)"<<endl;

cin>>col;

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

bla(x1,y1,x2,y2,col);

break;

}

default:

{

cout<<"Wrong choice";

}

}

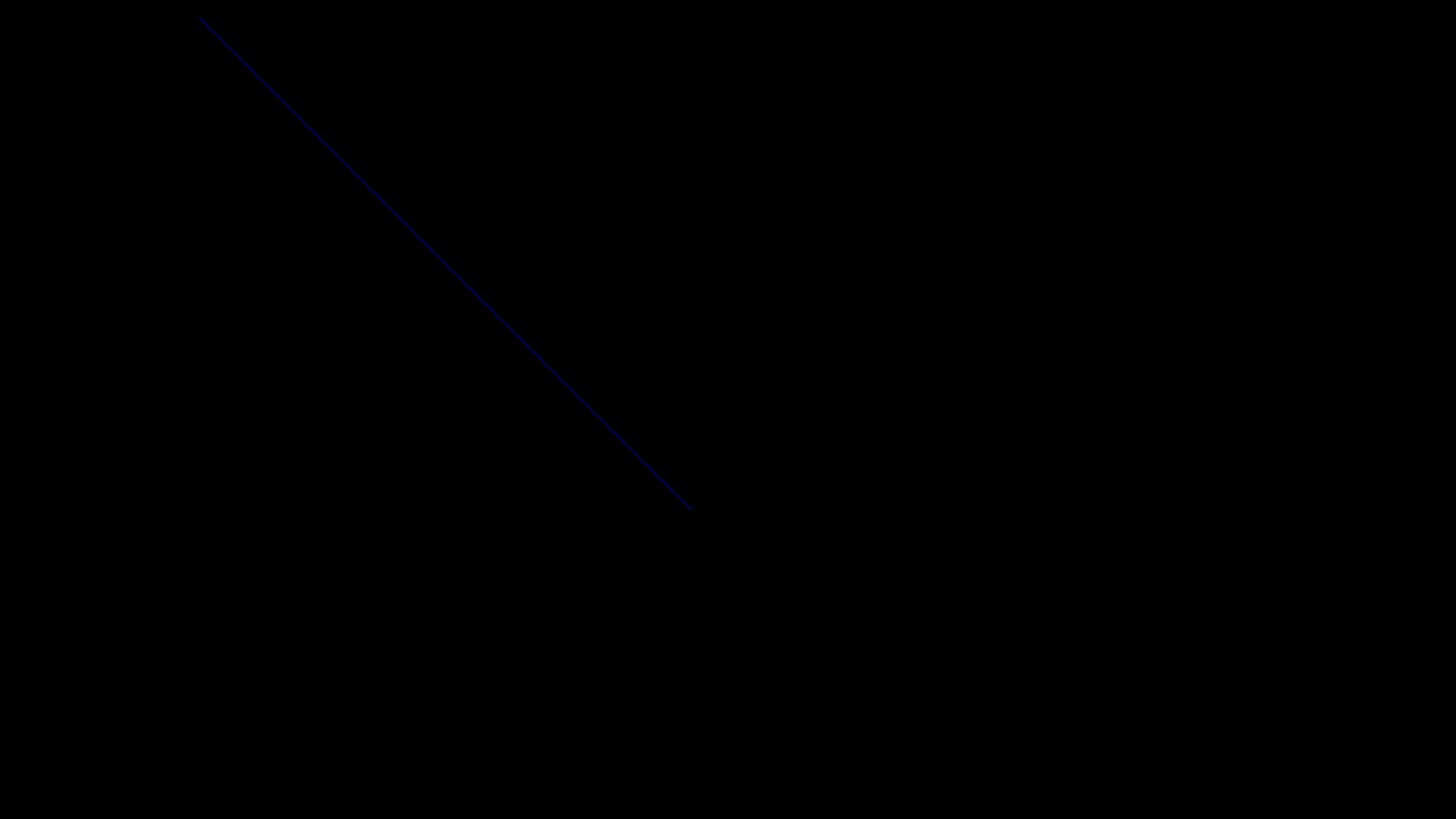
getch();

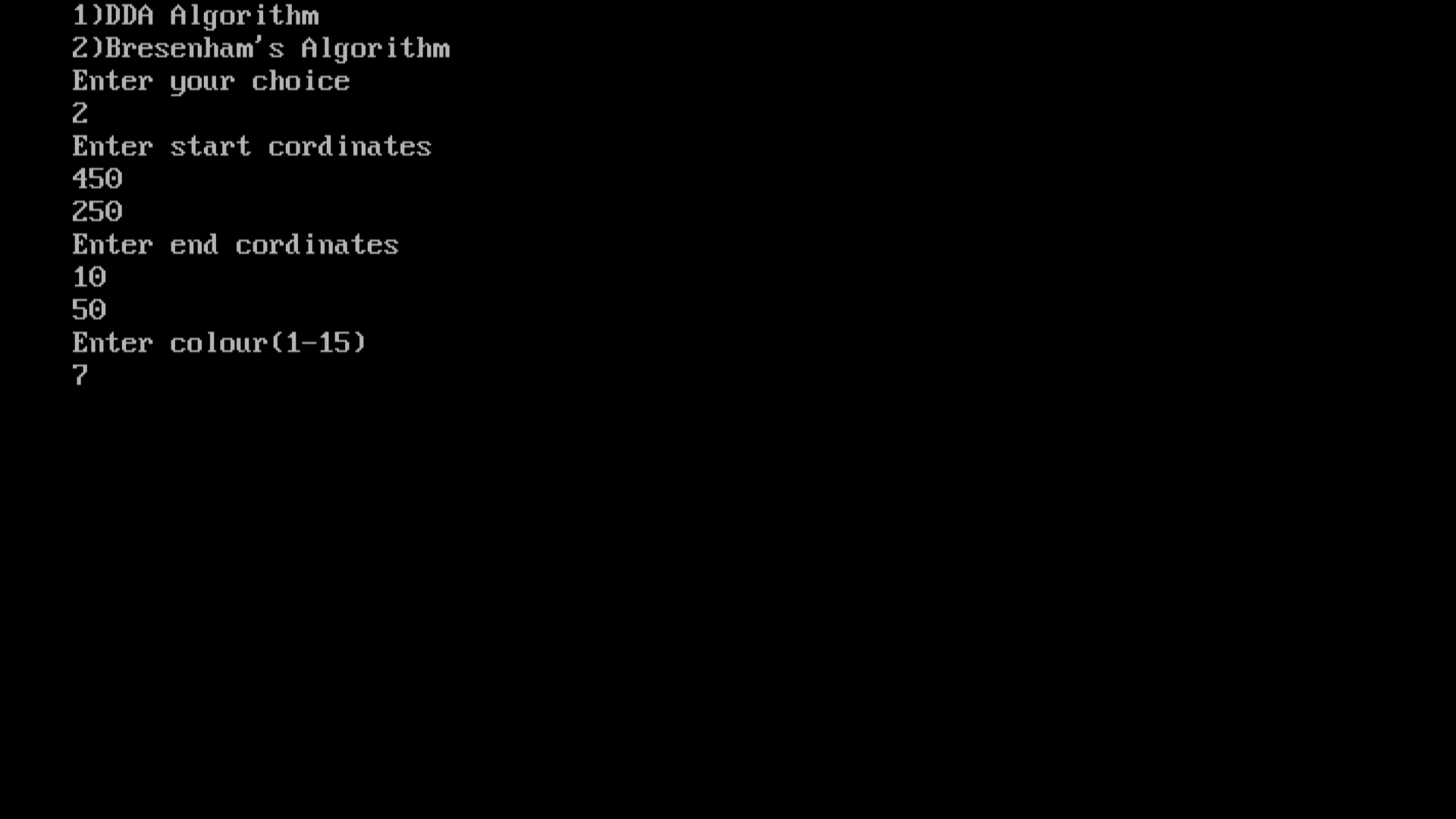
closegraph();

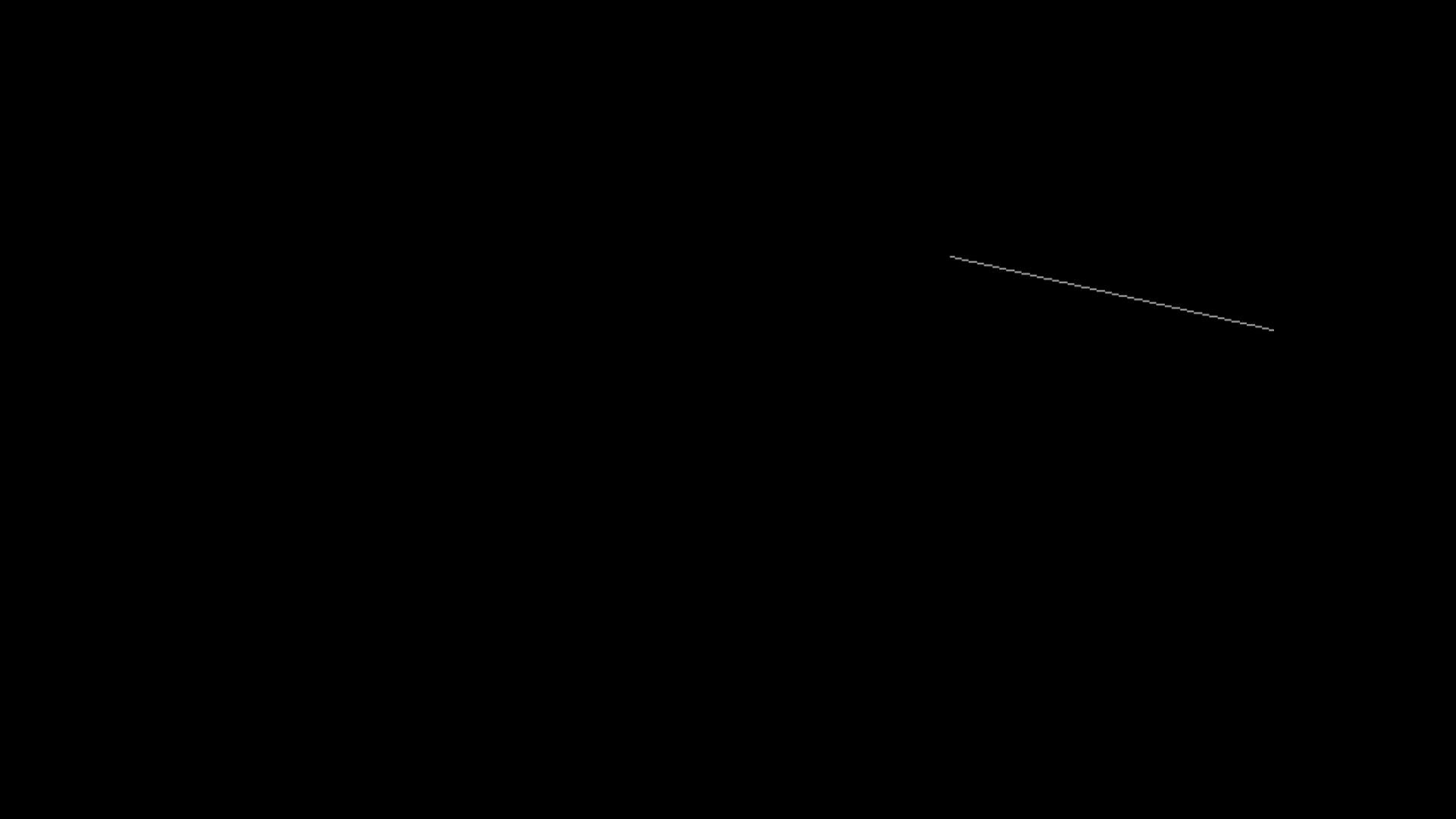
**Output:-**













**PROGRAM 5(Part-I)**

**Name:-** *Kaustubh S Kabra*

**Class:-** *Second Year Engineering Comp-1*

**Roll No:-** *20*

**Write a c++ program to draw pattern 1 using DDA Line and Bresenham Cricle Drwaing Algorithm**

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<iostream.h>

int sign(int x)

{

if(x>0)

return 1;

else if(x<0)

return -1;

else

return 0;

}

void dda(int x1,int y1,int x2,int y2)

{

float x,y,i,dx,dy,l;

if(x1==x2 && y1==y2)

{

putpixel(x1,y1,3);

}

else

{

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx>dy)

l=dx;

else

l=dy;

dx=(x2-x1)/l;

dy=(y2-y1)/l;

x=x1+0.5\*sign(dx);

y=y1+0.5\*sign(dy);

i=1;

while(i<l)

{

putpixel(x,y,3);

x=x+dx;

y=y+dy;

i++;

}

}

}

void show(int x1,int y1,int x,int y)

{

putpixel(x1+x,y1+y,4);

putpixel(x1-x,y1+y,4);

putpixel(x1+x,y1-y,4);

putpixel(x1-x,y1-y,4);

putpixel(x1+y,y1+x,4);

putpixel(x1-y,y1+x,4);

putpixel(x1+y,y1-x,4);

putpixel(x1-y,y1-x,4);

}

void b\_circle(int x1,int y1,int r)

{

int d;

d=3-2\*r;

int x=0,y=r;

show(x1,y1,x,y);

while(y>=x)

{

x++;

if(d>0)

{

y--;

d=d+4\*(x-y)+10;

}

else

{

d=d+4\*x+6;

}

show(x1,y1,x,y);

}

}

void main()

{

clrscr();

int x1,x2,y1,y2,col;

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

b\_circle(300,250,100);

b\_circle(300,250,50);

dda(300,150,385,300);

dda(300,150,215,300);

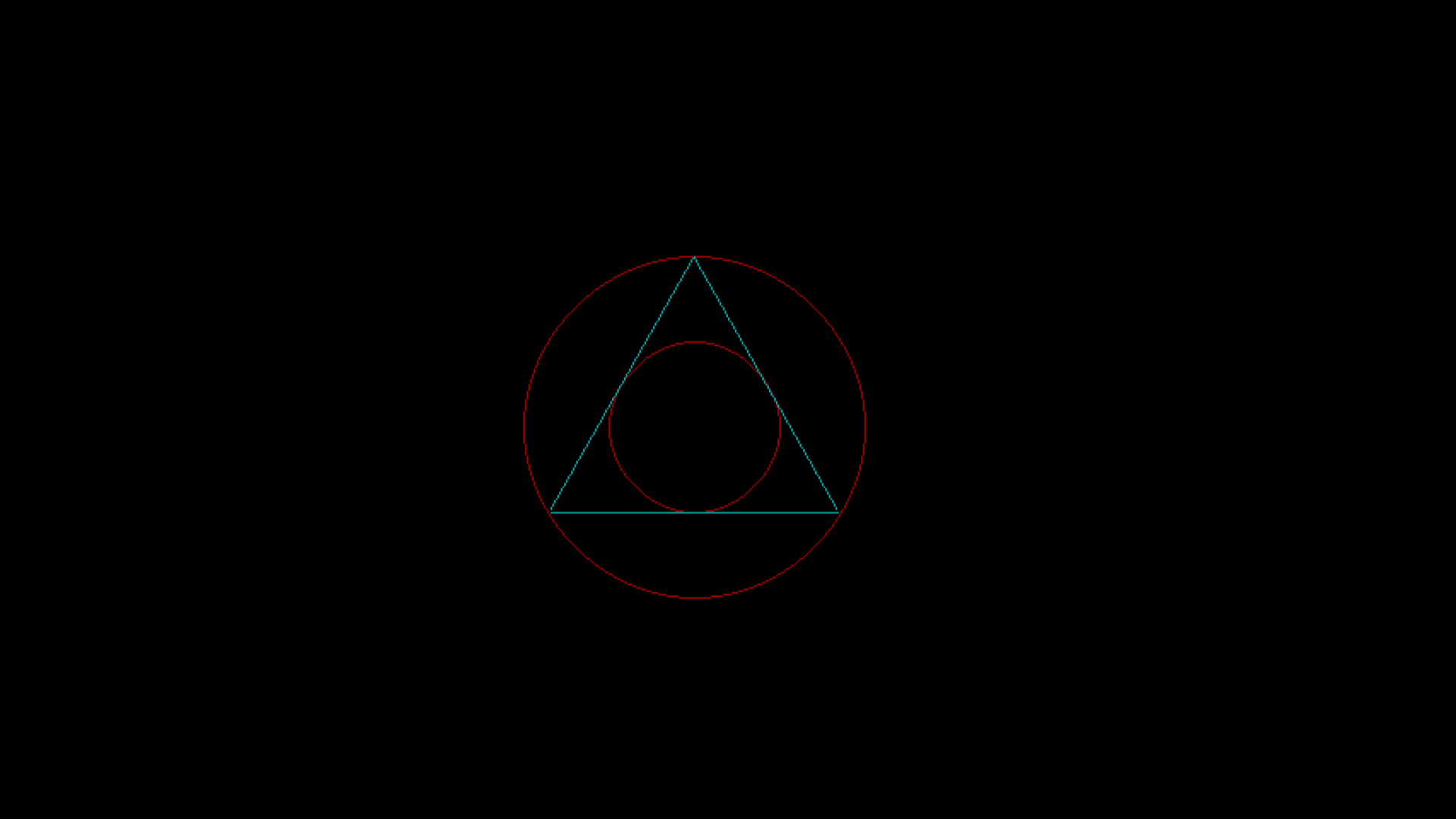
dda(385,300,215,300);

getch();

closegraph();

}

**Output:-**



**PROGRAM 5(Part-II)**

**Name:-** *Kaustubh S Kabra*

**Class:-** *Second Year Engineering Comp-1*

**Roll No:-** *20*

**Write a c++ program to draw pattern 2 using DDA Line and Bresenham Cricle Drwaing Algorith.**

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<iostream.h>

int sign(int x)

{

if(x>0)

return 1;

else if(x<0)

return -1;

else

return 0;

}

void dda(int x1,int y1,int x2,int y2)

{

float x,y,l,i,dx,dy;

if(x1==x2 && y1==y2)

{

putpixel(x1,y1,4);

}

else

{

dx=abs(x2-x1);

dy=abs(y2-y1);

if(dx>=dy)

l=dx;

else

l=dy;

dx=(x2-x1)/l;

dy=(y2-y1)/l;

x=x1+0.5\*sign(dx);

y=y1+0.5\*sign(dy);

i=1;

while(i<l)

{

putpixel(x,y,4);

x=x+dx;

y=y+dy;

i++;

}

}

}

void bla(int x1,int y1,int x2,int y2)

{

float dx,dy,x,y,e,i;

if(x1==x2 && y1==y2)

putpixel(x1,y1,4);

else

{

dx=abs(x2-x1);

dy=abs(y2-y1);

x=x1;

y=y1;

putpixel(x,y,4);

e=2\*dy-dx;

i=1;

while(i<=dx)

{

while(e>=0)

{

y=y+1;

e=e-2\*dx;

}

x=x+1;

e=e+2\*dy;

putpixel(x,y,4);

i=i+1;

}

}

}

void main()

{

clrscr();

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

bla(200,300,400,300);

dda(200,300,200,200);

bla(200,200,400,200);

dda(400,200,400,300);

dda(200,250,300,200);

dda(200,250,300,300);

bla(300,200,400,250);

dda(300,300,400,250);

bla(250,225,150,225);

dda(250,225,250,275);

dda(350,275,250,275);

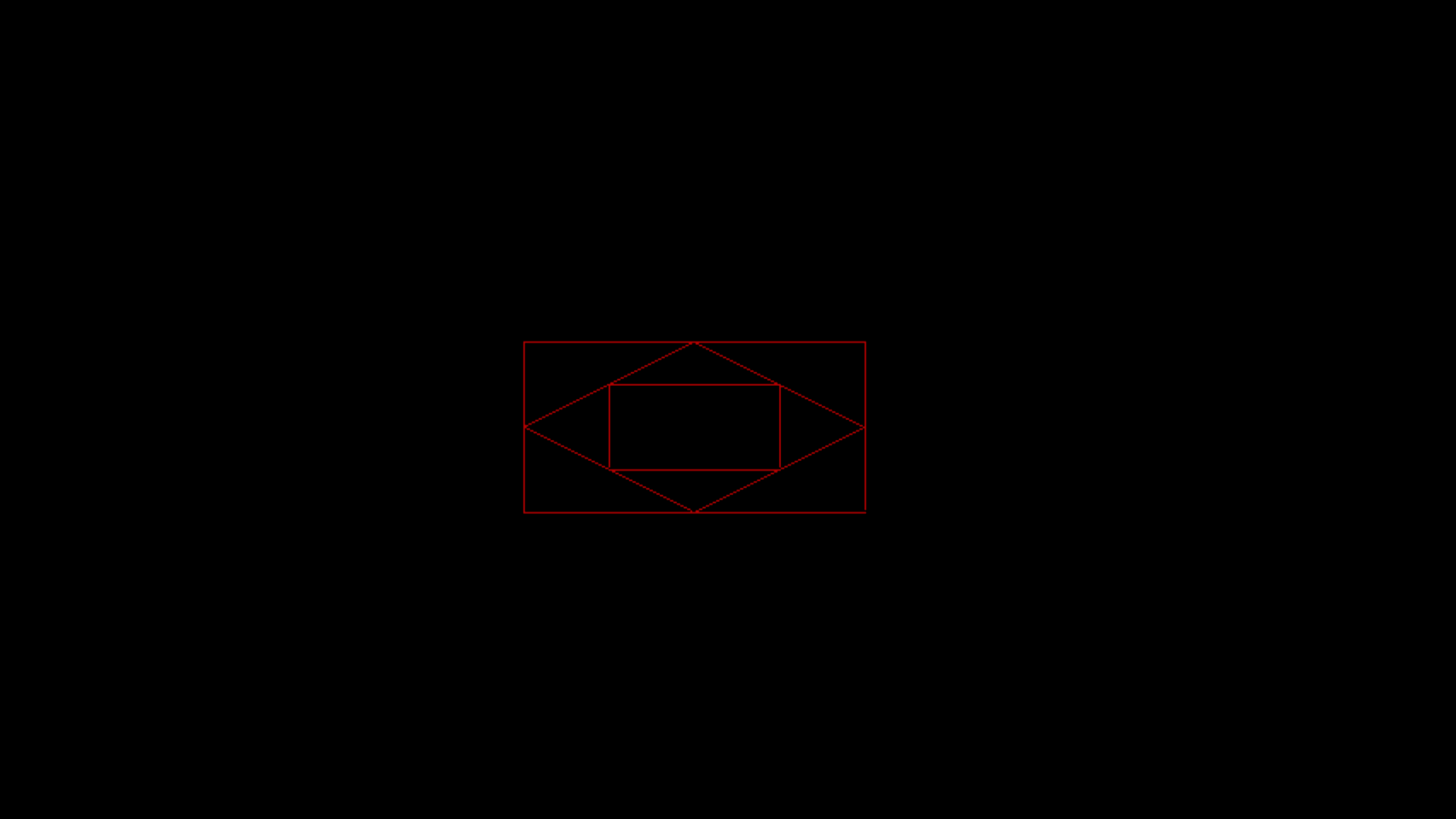
dda(350,225,350,275);

getch();

**closegraph();**

**}**

**Output:-**



**PROGRAM 6**

**Name:-** *Kaustubh S Kabra*

**Class:-** *Second Year Engineering Comp-1*

**Roll No:-** *20*

**Write a c++ program to draw a concave polygon and fill it with desired color using scan fill algorithm.**

#include<graphics.h>

#include<conio.h>

#include<iostream.h>

#include<dos.h>

void main()

{

clrscr();

int gd=DETECT,gm,dx,dy,x,y,temp,n,i,j,k;

int p[20][2],xi[20];

float slope[20];

cout<<"Enter total number of vertices of the polygon: ";

cin>>n;

cout<<"Enter x and y cordinates of the vertices: "<<endl;

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

{

cout<<"x"<<i<<"y"<<i<<" ";

cin>>p[i][0]>>p[i][1];

}

p[n][0]=p[0][0];

p[n][1]=p[0][1];

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

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

{

line(p[i][0],p[i][1],p[i+1][0],p[i+1][1]);

}

getch();

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

{

dx=p[i+1][0]-p[i][0];

dy=p[i+1][1]-p[i][1];

if(dy==0)

{

slope[i]=1.0;

}

if(dx==0)

{

slope[i]=0.0;

}

if((dy!=0) && (dx!=0))

{

slope[i]=(float) dx/dy;

}

}

for(y=480;y>0;y--)

{

k=0;

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

{

if(((p[i][1]<=y)&&(p[i+1][1]>y))||((p[i][1]>y)&&(p[i+1][1]<=y)))

{

xi[k]=(int)(p[i][0]+slope[i]\*(y-p[i][1]));

k++;

}

}

for(j=0;j<k-1;j++)

for(i=0;i<k-1;i++)

{

if(xi[i]>xi[i+1])

{

temp=xi[i];

xi[i]=xi[i+1];

xi[i+1]=temp;

}

}

setcolor(11);

for(i=0;i<k;i+=2)

{

line(xi[i],y,xi[i+1],y);

delay(20);

}

}

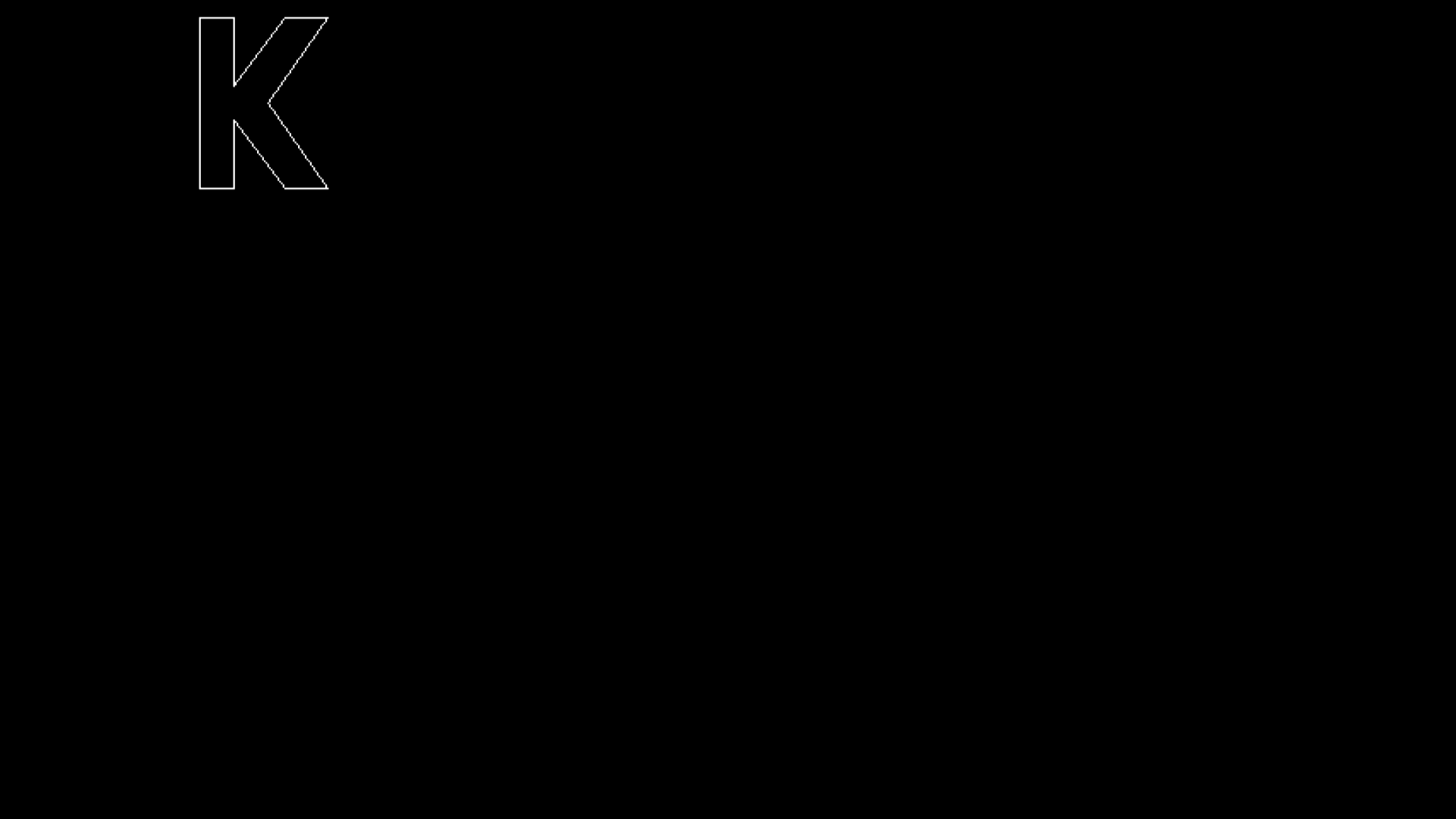
getch();

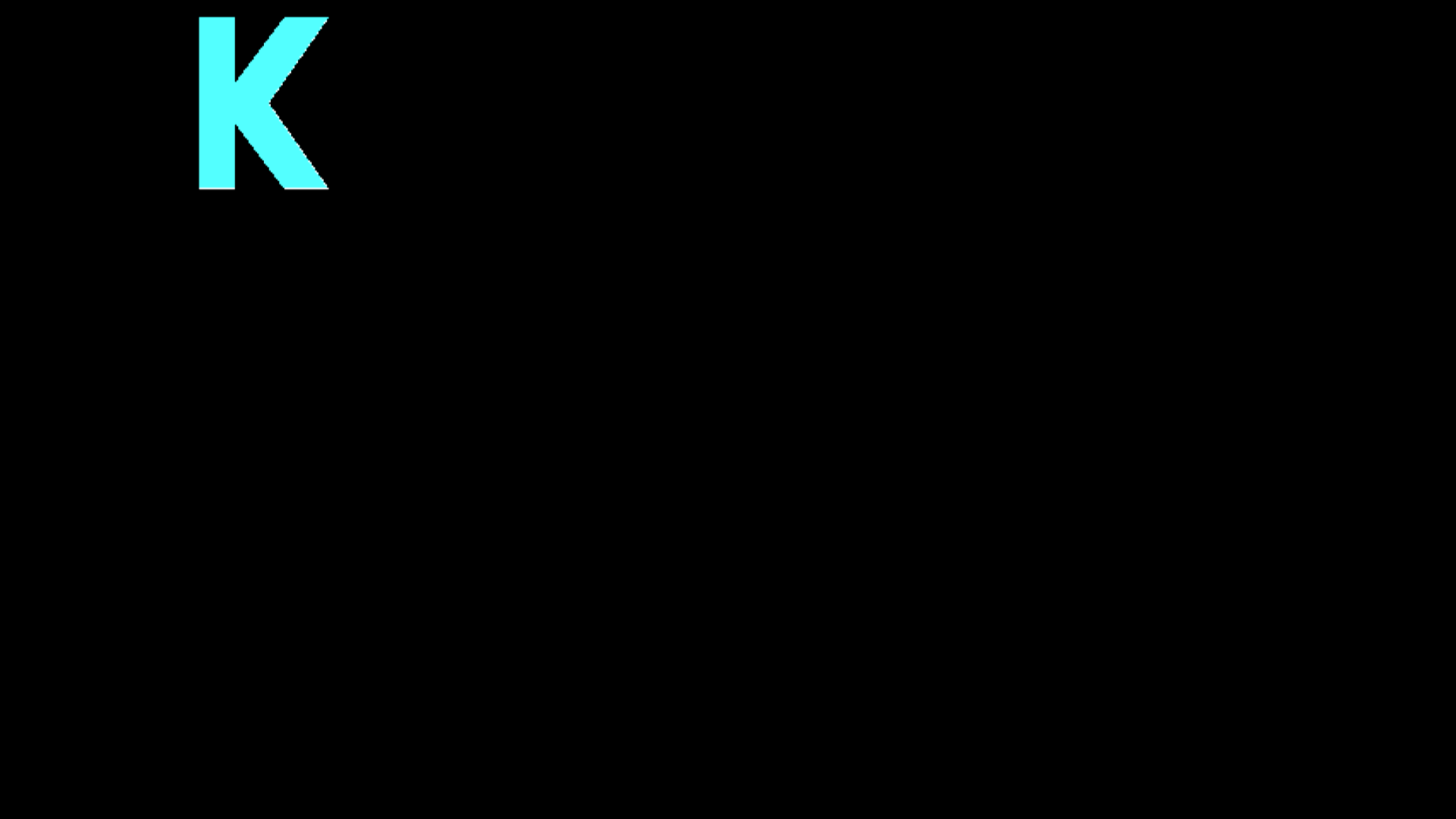
closegraph();

}

**Output:-**







**PROGRAM 7**

**Name:-** *Kaustubh S Kabra*

**Class:-** *Second Year Engineering Comp-1*

**Roll No:-** *20*

**Write a c++ program to implement Cohen Southerland line clipping algorithm.**

#include<graphics.h>

#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

static int LEFT=1,RIGHT=2,BOTTOM=4,TOP=8,xmax,ymax,xmin,ymin;

int find\_code(int x,int y)

{

int code=0;

if(y>ymax)

code|=TOP;

if(y<ymin)

code|=BOTTOM;

if(x>xmax)

code|=RIGHT;

if(x<xmin)

code|=LEFT;

return code;

}

void main()

{

clrscr();

int x1,y1,x2,y2;

int gd=DETECT,gm;

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

setcolor(CYAN);

cout<<"Enter maximum and minimum value of window: ";

cin>>xmin>>ymin>>xmax>>ymax;

rectangle(xmin,ymin,xmax,ymax);

cout<<"Enter start (x1,y1) and end points (x2,y2) of the line: ";

cin>>x1>>y1>>x2>>y2;

line(x1,y1,x2,y2);

getch();

int ocode1=find\_code(x1,y1),ocode2=find\_code(x2,y2);

int accept=0;

while(1)

{

float m=(float)(y2-y1)/(x2-x1);

if(ocode1==0 && ocode2==0)

{

accept=1;

break;

}

else if((ocode1&ocode2)!=0)

{

break;

}

else

{

int x,y;

int temp;

if(ocode1==0)

{

temp=ocode2;

}

else

{

temp=ocode1;

}

if(temp&TOP)

{

x=x1+(ymax-y1)/m;

y=ymax;

}

else if(temp&BOTTOM)

{

x=x1+(ymin-y1)/m;

y=ymin;

}

else if(temp&LEFT)

{

x=xmin;

y=y1+m\*(xmin-x1);

}

else if(temp&RIGHT)

{

x=xmax;

y=y1+m\*(xmax-x1);

}

if(temp==ocode1)

{

x1=x;

y1=y;

ocode1=find\_code(x1,y1);

}

else

{

x2=x;

y2=y;

ocode2=find\_code(x2,y2);

}

}

}

setcolor(RED);

cout<<"After clipping";

if(accept)

{

line(x1,y1,x2,y2);

rectangle(xmin,ymin,xmax,ymax);

}

getch();

closegraph();

}

**Output:-**



