Shuvrojit Biswas

id : 180221011

***Course Title:* <u>Computer Graphics</u>**

[redacted]

Program to draw an ellipse using Midpoint Ellipse Algorithm

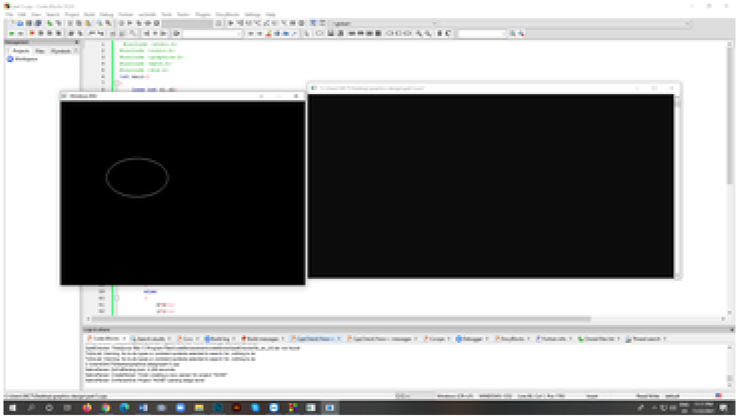
<u>Code:</u>

```
 #include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <dos.h>
int main()
{
    long int d1,d2;
    int i,gd=DETECT,gm,x,y;
    long int rx=80,ry=50,rxsq,rysq,tworxsq,tworysq,dx,dy;
    //printf("Enter the x Radius of the ellipse");
    // scanf("%ld",&rx);

    //printf("Enter the y Radius of the ellipse");
    //scanf("%ld",&ry);
    initgraph(&gd,&gm," ");
    rxsq=rx*rx;
    rysq=ry*ry;
    tworxsq=2*rxsq;
    tworysq=2*rysq;
    x=0;
    y=ry;
    d1=rysq - (rxsq * ry) + (0.25 * rxsq);
    dx= tworysq * x;
    dy= tworxsq * y;
    do
    {
        putpixel(200+x,200+y,15);
        putpixel(200-x,200-y,15);
    putpixel(200+x,200-y,15);
    putpixel(200-x,200+y,15);
    if (d1 < 0)
    {
        x=x+1;
        y=y;
        dx=dx + tworysq;
        d1=d1 + dx + rysq;
    }
    else
    {
        x=x+1;
        y=y-1;
        dx= dx + tworysq;
        dy= dy - tworxsq;
        d1= d1 + dx - dy + rysq;
    }
    delay(50);
}
while (dx < dy);
d2 = rysq * ( x + 0.5) * ( x + 0.5 ) + rxsq * (y - 1) * (y-1) - rxsq * rysq;
do
{
    putpixel(200+x,200+y,15);
    putpixel(200-x,200-y,15);
    putpixel(200+x,200-y,15);
    putpixel(200-x,200+y,15);

    if (d2 >0)
    {
        x=x;
        y=y-1;
        dy = dy - tworxsq;
        d2 = d2 - dy + rxsq;
    }
    else
    {
            x= x+1;
            y=y-1;
            dy=dy - tworxsq;
            dx= dx + tworysq;
            d2 = d2 + dx -dy + rxsq;
        }
        delay(50);
    }
    while ( y> 0);
    getch();
    closegraph();
}
```

Screenshot of the console:



Part 6.A: To implement 4-connected flood fill algorithm

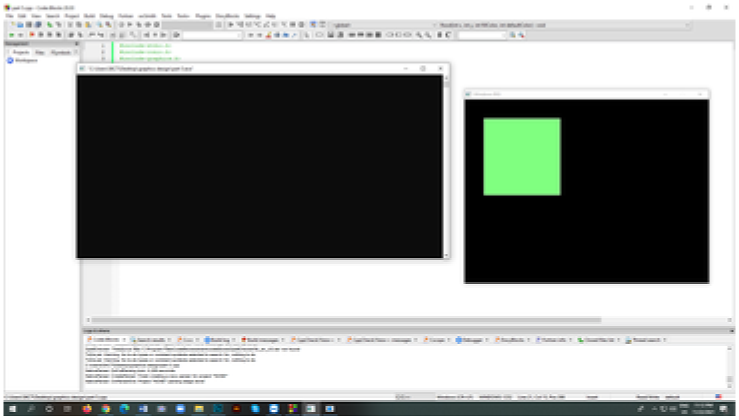
Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>

void flood(int,int,int,int);
```

```
int main()
{
   int gd=DETECT,gm;
   initgraph(&gd,&gm,"C:/TURBOC3/bgi");
   rectangle(50,50,250,250);
   flood(55,55,10,0);
   getch();
}
void flood(int x,int y,int fillColor, int defaultColor)
{
   if(getpixel(x,y)==defaultColor)
   {
      delay(1);
      putpixel(x,y,fillColor);
      flood(x+1,y,fillColor,defaultColor);
      flood(x-1,y,fillColor,defaultColor);
      flood(x,y+1,fillColor,defaultColor);
      flood(x,y-1,fillColor,defaultColor);
   }
}
```

Screenshot of the console:

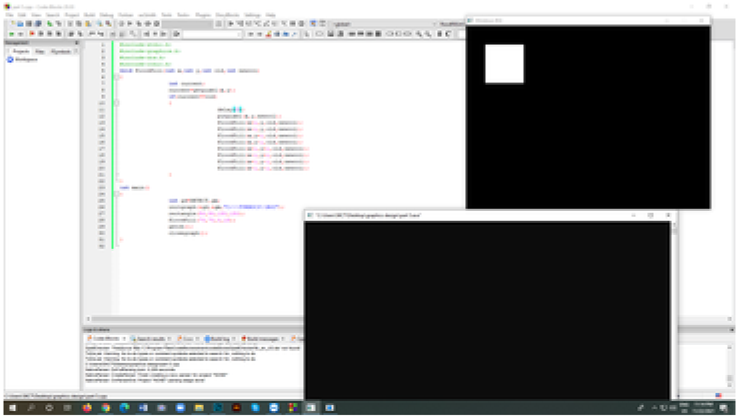


Part 6.B: To implement 8-connected flood fill algorithm

Code:

```
#include<stdio.h>
#include<graphics.h>
#include<dos.h>
#include<conio.h>
void floodfill(int x,int y,int old,int newcol)
{
	int current;
	current=getpixel(x,y);
	if(current==old)
	{
		delay(5);
		putpixel(x,y,newcol);
		floodfill(x+1,y,old,newcol);
		floodfill(x-1,y,old,newcol);
		floodfill(x,y+1,old,newcol);
		floodfill(x,y-1,old,newcol);
		floodfill(x+1,y+1,old,newcol);
		floodfill(x-1,y+1,old,newcol);
		floodfill(x+1,y-1,old,newcol);
		floodfill(x-1,y-1,old,newcol);
	}
}
int main()
{
	int gd=DETECT,gm;
	initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
	rectangle(50,50,150,150);
	floodfill(70,70,0,15);
	getch();
	closegraph();
}
```

Screenshot of the console:



PART 9.A : TO IMPLEMENT POINT CLIPPING

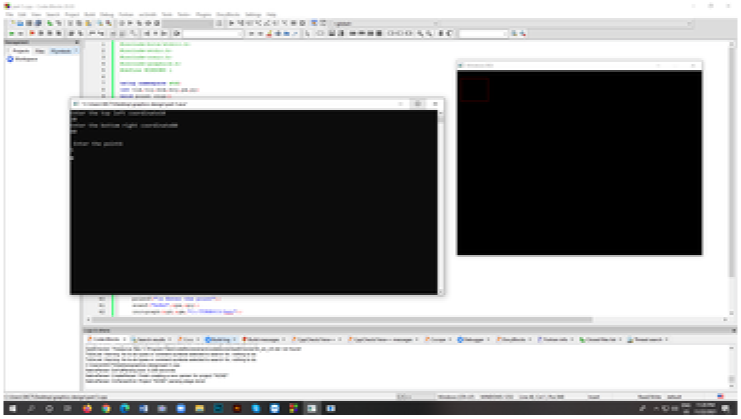
Code:

```
#include<bits/stdc++.h>
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#define WINDOWS 1

using namespace std;
int tlx,tly,brx,bry,px,py;
void point_clip()
{
   int wxmin,wymin,wxmax,wymax;
   wxmin=tlx;
   wxmax=brx;
   wymin=tly;
   wymax=bry;
   if(px>=wxmin&&px<=wxmax)
      if(py>=wymin&&py<=wymax)
         putpixel(px,py,RED);
   getch();
   closegraph();
}
```

```
void clrscr() {
 #ifdef WINDOWS
 system("cls");
 #endif
 #ifdef LINUX
 system("clear");
 #endif
}
int main()
{
   int gd=DETECT,gm,xc,yc,r;
   clrscr();
   printf("Enter the top left coordinate");
   scanf(" %d %d",&tlx,&tly);
   printf("Enter the bottom right coordinate");
   scanf("%d%d",&brx,&bry);
   printf("\n Enter the point");
   scanf("%d%d",&px,&py);
   initgraph(&gd,&gm,"C:/TURBOC3/bgi");
   setbkcolor(BLUE);
   setcolor(RED);
   rectangle(tlx,tly,brx,bry);
   point_clip();
}
```

Screenshot of the console:



```
Enter the top left coordinate
[illegible]
Enter the bottom right coordinate
[illegible]

 Enter the point
[illegible]
```

Part 9.B: Program to perform Line Clipping using Cohen Sutherland Algorithm

Code:

```
#include <iostream>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
using namespace std;
class data
{
   int gd, gmode, x, y, xmin,ymin,ymax,xmax;
   int a1,a2;
   float x1, y1,x2,y2,x3,y3;
   int xs, ys, xe, ye;
   float maxx,maxy;
public:
   void getdata ();
   void find ();
   void clip ();
   void display (float, float,float,float);
   void _Fazle Rabbi(int);
   void showbit (int);
};




void data :: getdata ()
{
   cout<<"Enter the minimum and maximum coordinate of window (x, y) ";
   cin >>xmin>>ymin>>xmax>>ymax;
   cout<<"Enter the end points of the line to be clipped";
   cin >>xs>>ys>>xe>>ye;
   display (xs, ys, xe,ye);
```

```
}
void data :: display (float xs, float ys,float xe, float ye)
{
    int gd=DETECT;
    initgraph (&gd,&gmode, "");
    maxx=getmaxx();
    maxy=getmaxy();
    line (maxx/2,0,maxx/2,maxy);
    line (0, maxy/2,maxx,maxy/2);
    rectangle (maxx/2+xmin,maxy/2-ymax,maxx/2+xmax,maxy/2-ymin);
    line (maxx/2+xs,maxy/2-ys,maxx/2+xe,maxy/2-ye);
    getch();
}
void data :: find ()
{
    a1=0;
    a2=0;
    if ((ys-ymax)>0)
        a1+=8;
    if ((ymin-ys)>0)
        a1+=4;
    if ((xs-xmax)>0)
        a1+=2;
    if ((xmin-xs)>0)
        a1+=1;
    if ((ye-ymax)>0)
        a2+=8;
    if ((ymin-ye)>0)
        a2+=4;
    if ((xe-xmax)>0)
        a2+=2;
    if ((xmin-xe)>0)
        a2+=1;
    cout<<"\nThe area code of 1st point is ";
    showbit (a1);
    getch ();
    cout <<"\nThe area code of 2nd point is ";
    showbit (a2);
    getch ();
}
void data :: showbit (int n)
{
    int k,p;
    for (int i=3; i>=0; i--)
    {
        p =1<<i;
        //k = n?
        k ==0?cout<<"0": cout<<"1";
    }
}
void data ::clip()
{
    int j=a1&a2;
    if (j==0)
    {
        cout<<"\nLine is perfect candidate for clipping";
        if (a1==0)
        {


            _sorna(a1);
            x2=x1;
            y2=y1;

            if (a2=0)
            {
                x3=xe;
                y3=ye;
            }
            else
            {
                _Fazle Rabbi(a2);
                x3=x1;
                y3=y1;
            }
            xs=x2;
            ys=y2;
            xe=x3;
            ye=y3;
            cout << endl;
            display (xs,ys,xe,ye);
            cout<<"Line after clipping";
            getch ();
        }
        else if ((a1==0) && (a2=0))
        {
            cout <<"\n Line is in the visible region";
            getch ();
        }
    }
}
    void data :: _sorna(int i)
    {
        int j, k,l,m;
        i=i&1;
        x1=0;
        y1=0;
        if (1==1)
        {
            x1=xmin;
            y1=ys+ ((x1-xs)/ (xe-xs))*(ye-ys);
        }
        j=i&8;
        if (j>0)
        {
            y1=ymax;
            x1=xs+(y1-ys)/((ye-ys))*(xe-xs);
        }
        k=i & 4;
        if (k==1)
        {
          y1=ymin;
          x1=xs+((y1-ys)/(ye-ys))*(xe-xs);
       }
       m= i&2;
       if (m==1)
       {
          x1=xmax;
          y1=ys+ ((x1-xs)/ (xe-xs))*(ye-ys);
       }

    }

int main ()
       {
          data s;
         // clrscr();
          s.getdata();
          s.find();
          getch();
          closegraph ();
          return 0;
       }
```

Screenshot of the console: