INSTRUCTION MANUAL

COMPUTER GRAPHICS LAB

(BTCS-509)

Prepared by

Er. Diana
Assistant Professor (CSE)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE

LUDHIANA – 141006

DECLARATION

This Manual of Computer Graphics Lab (BTCS-509) has been prepared by me as per syllabus of Computer Graphics Lab (BTCS-509).

Signature

SYLLABUS

1. To plot a point (pixel) on the screen.
2. To draw a straight line using DDA Algorithm.
3. To draw a straight line using Bresenham's Algorithm.
4. Implementation of mid-point circle generating Algorithm.
5. Implementation of ellipse generating Algorithm.
6. To translate an object with translation parameters in X and Y directions.
7. To scale an object with scaling factors along X and Y directions.
8. To rotate an object with a certain angle about origin.
9. Perform the rotation of an object with certain angle about an arbitrary point.
10. To perform composite transformations of an object.
11. To perform the reflection of an object about major axis.
12. Write a program for flood fill algorithm.
13. Write a program to fill for connected area of pixel by using boundary filling algorithm.

\

INDEX

PRACTICAL 1

AIM: To plot a point(pixel) on screen.

Objective :- To Draw a point with given parameters using a suitable Programming Language.

Program :-

```
#include <iostream.h >

#include < conio.h >

#include < graphics.h >

void main()

{
      int gdriver = DETECT, gmode;
      initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
      putpixel(100,100,15);
      getch();
}
```

OUTPUT:-

PRACTICAL 2

AIM: To draw a straight line using DDA Alogrithm.

Objective :- To Draw a Line with given parameters using a suitable Programming Language.

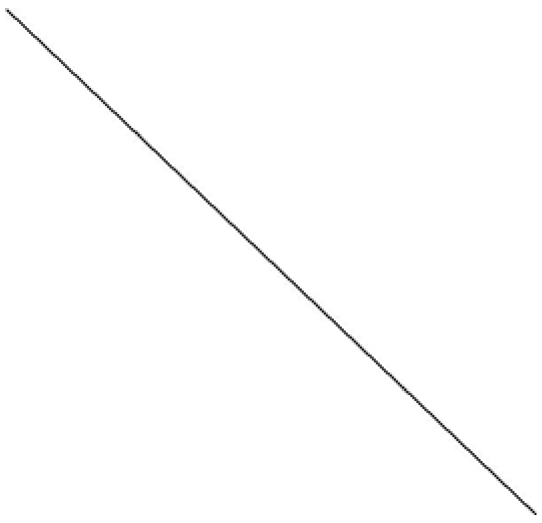
Program :-

```
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
#include<math.h>

#define ROUND(a)((int)(a+0.5))

void DDA(int xa,int ya,int xb,int yb)

{
    int dx=xb-xa,dy=yb-ya,steps,k;
    float xinc,yinc,x=xa,y=ya;
    if(abs(dx)>abs(dy))
    steps=abs(dx);

else


    steps=abs(dy);
    xinc=dx/float(steps);
    yinc=dy/float(steps);

putpixel(ROUND(x),ROUND(y),15);

for(k=0;k<steps;k++)

{
    x=x+xinc;
    y=y+yinc;
    putpixel(ROUND(x),ROUND(y),15);
}}
void main()
```

```
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"c:\tc:\bgi");
    int xa,xb,ya,yb;
    cout<<"enter the points";
    cin>>xa>>xb>>ya>>yb;
    DDA(xa,xb,ya,yb);
getch();
  }
```

Output :-

```
enter starting coordinates of line: 50 60
enter ending coordinates of line : 300 250
```



PRACTICAL 3

AIM: To draw a straight line using Bresenham’s Alogrithm.

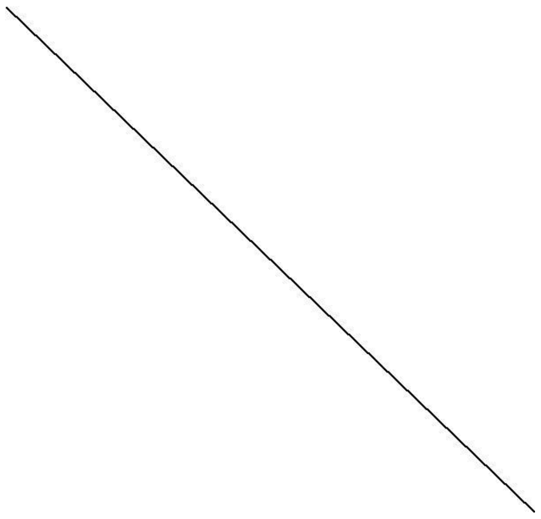
Objective :- To Draw a Line with given parameters using a suitable Programming Language.

Program :-

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void lineBres(int xa,int ya,int xb,int yb)
{
   int dx=abs(xa-xb),dy=abs(ya-yb);
   int p=2*dy-dx;
   int twoDy=2*dy,twoDyDx=2*(dy-dx);
   int x,y,xEnd;
   clrscr();
//determine which point to use as start,which as end
   if(xa>xb)
   {
     x=xb;
     y=yb;
     xEnd=xa;
   }
else
  { x=xa;
   y=ya;
   xEnd=xb;
}
putpixel(x,y,45);
   while(x<xEnd)
   {
    x++;
    if(p<0)
    p+=twoDy;
else
{y++;

+=twoDyDx;
}
 putpixel(x,y,45);
}}
 void main()
 {
    int gd=DETECT,gm;
    initgraph(&gd,&gm," c:\tc:\bgi");
    int xa,ya,xb,yb;
    cout<<"enter the points";
    cin>>xa>>ya>>xb>>yb;
    lineBres(xa,ya,xb,yb);
    getch();
 }
```

Output :-

```
enter starting coordinates of line: 40 40
enter ending coordinates of line : 300 300
```



PRACTICAL 4

AIM: Implementation of mid-point circle generating Algorithm.

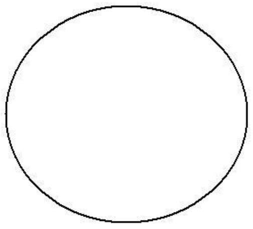
Objective:- To Draw a circle with given parameters using a suitable Programming Language.

Program:-

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>

void bcircle(int xcent,int ycent,int x,int y);

void main()

{
   int gd=DETECT,gm;
   initgraph(&gd,&gm," ");
   int x=0,y,d,r,xcent=300,ycent=300;
   cout<<"Enter the radius";
   cin>>r;
   y=r;
   d=5/4-r;

 while(x<y)
 {
    bcircle(xcent,ycent,x,y);
    if(d<0)
    {
      x=x+1;
      d=d+2*x+1;
    }
    else
    {
      x=x+1;
      y=y-1;
      d=d+2*(x-y)+1;
    }
  }

Getch();
}
void bcircle(int xcent,int ycent,int x,int y)

{
putpixel(xcent-x,ycent-y,6);
putpixel(xcent-y,ycent-x,12);
putpixel(xcent+y,ycent-x,24);
putpixel(xcent+x,ycent-y,14);
putpixel(xcent+x,ycent+y,13);
putpixel(xcent+y,ycent+x,9);
putpixel(xcent-y,ycent+x,27);
putpixel(xcent-x,ycent+y,5);
getch();
closegraph();
}
```

<u>Output</u> :-

```
enter the values for x and y: 250 300

enter the value for radius:60
```



PRACTICAL 5

AIM: Implementation of ellipse generating algorithm

Objective :- To Draw a ellipse with given parameters using a suitable Programming Language.

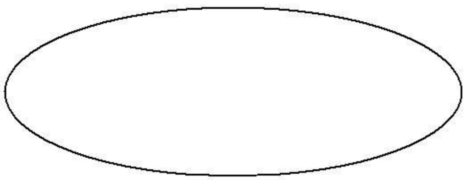
Program:-

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
#include<math.h>

int xc,yc,rx,ry;
void main()
{

        void plotpoints(int,int);
        int round(float);
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"");
        cout<<"enter centre point";
        cin>>xc>>yc; cout<<"enter
        major radius"; cin>>rx;

        cout<<"enter minor radius";
        cin<<ry;
        long int p,px,py,x,y;
        long int ry2=ry*ry;
        long int ry22=rx*rx;
        long int rx=rx2*2;
        x=0;
        y=ry;
        plotpoints(x,y); P=round(ry2-
        rx2*ry+(2.25*rx2)); Px=0;

        Py=rx22*y;
        while(px<py)
        {

            x++;
            px=px+ry22;

         if(p>=0)

         {

            y--; py=py-
            rx22;

          }

            if(p<0)
               P=p+px+ry2;
          else
               P=p+px=ry2-py;
          Plotpoints(x,y);

         }

            P=round(ry2*(x+0.5)*(x+0.5)+rx2*(y-1)*(y-1)-rx2*ry2);
       While(y>0)

       {
          y--; py=py-
          rx22;
          if(p<=0)

       {
          x++;
          px=px+ry22;
       }
       if(p>=0) P=p-
          px+rx2;
        else P=p+px-
          py=rx2;

        Plotpoints(x,y);
       }
       getch();
       closegraph();
}
void plotpoints(int x,int y)
```

```
          {
          setlinestyle(0,0,3);
          line(xc+x,yc+y,xc+x,yc+y);
          line(xc-x,yc+y,xc-x,yc+y);
          line(xc+x,yc-y,xc+x,yc-y);
          line(xc-x,yc-y,xc-x,yc-y);
}
         int round(float x)
{
       int y=avs(x);
       if(x>(y+0.5))
          return(y+1);

       else

           return(y);

}
```

<u>Output</u> :-

```
Enter center point
200 250

enter major radius
70
enter minor radius
30
```



PRACTICAL 6

AIM: To translate an object with translation parameters in X and Y directions.

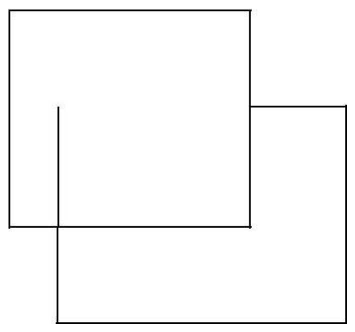
Objective :- To Change the Coordinates of a given Object.

Program:-

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void main()
{
            int gd=DETECT,gm;
             int x1,y1,x2,y2;
             initgraph(&gd,&gm,"");
            cout<<"Enter the starting coordinates of rectangle":
            cin>>x1>>y1;
            cout<<"enter the coordinates of rectangle:";
               cin>>x2>>y2;
             rectangle(x1,y1,x2,y2);
            cout<<"enter the translational coordinates:";
            cin>>tx>>ty;
            rectangle(x1+tx,y1+ty,x2+tx,y2+ty); getch();
            closegraph();
}
```

Output :-

```
enter the starting coordinates of rectangle:
100 150

enter the ending coordinates of rectangle:
250 300

enter the translation coordinates: 40 60
```



PRACTICAL 7

AIM: To scale an object with scaling factors along X and Y directions.

Objective :- To alter the size of an object.

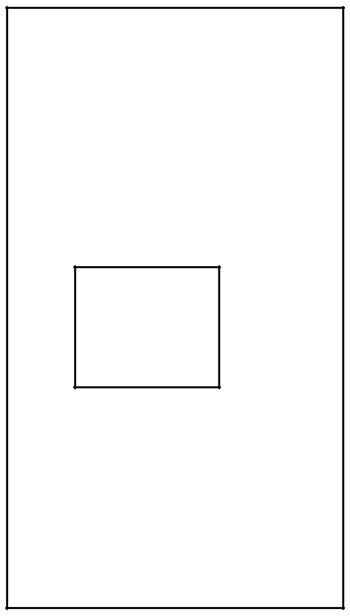
Program:-

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void main()
{

        int gd=DETECT,gm;
        initgraph(&gd,&gm,”c:\tc:\bgi”); int
        x1=100,y1=100,x2=200,y2=200;
        cleardevice(); rectangle(x1,y1,x2,y2);

        int sx,sy,xf,yf;
        cout<<”enter fixed points x and y;
        cin>>xf>>yf;
       cout<<”enter scaling factors sxa nd sy”;
        cin>>sx>>sy;
        int xa=(x1*sx)+(xf*(1-sx));
        int ya=(y1*sy)+(yf*(1-sy));
        int xb(x2*sx)+(xf*(1-sx)); int
        yb=(y2*sy)+(xf*(1-sy));
        rectangle(xa,ya,xb,yb);
        getch();
          closegraph();
}
```

Output :-

```
enter fixed points x and y
120 140
```

```
enter scaling factors sx and
sy
2 3
```



PRACTICAL 8

AIM: To rotate an object with a certain angle about origin.

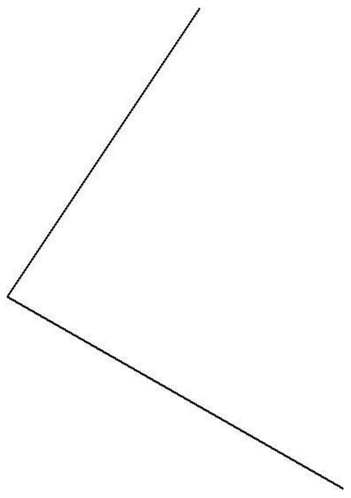
Objective :- To repositioning a given object along a circular path in the xy plane.

Program:-

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>

void main()
{

          int gd=DETECT,gm; int
          x1,y1,x2,y2;
          initgraph(&gd,&gm,"");
         cout<<"Enter value for x1,1";
         cin>>x1>>y1;
         cout<<"enter the value for x2,y2";
         cin>>x2>>y2;
          line(x1,y1,x2,y2);
          int rx,ry,x,y,q;
         cout<<"enter rotational coordinates";
         cin>>rx>>ry;
         cout<<"enter engle";
         cin>>q;
          double ang=double"(q)*(3.142/180);
          int xa,yb,xb,yb;
         xa=rx+((x1-rx)*cos(ang))-((y1-ry)*sin(ang));
           ya=ry+((x1-rx)*sin(ang))-((y1-ry)*cos(ang));
         xb=rx+((x2-rx)*cos(ang)-((y2-ry)*sin(ang));
           yb=ry+((x2-rx)*sin(ang))-((y2-ry)*cos(ang));
          line(xa,ya,xb,yb);
          getch();
         closegraph();

}
```

utput :-

```
nter value for x1,y1: 200 300

nter value for x2,y2: 250 150

nter rotational coordinates: 200 300
nter angle 70
```



PRACTICAL 9

AIM: To perform reflection of an object about major axis.

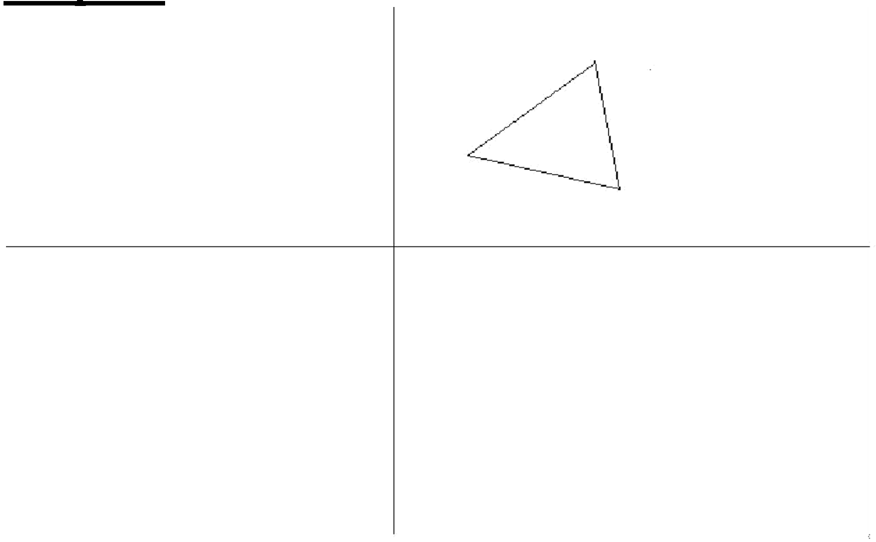
Objective:- This program perform reflection here.

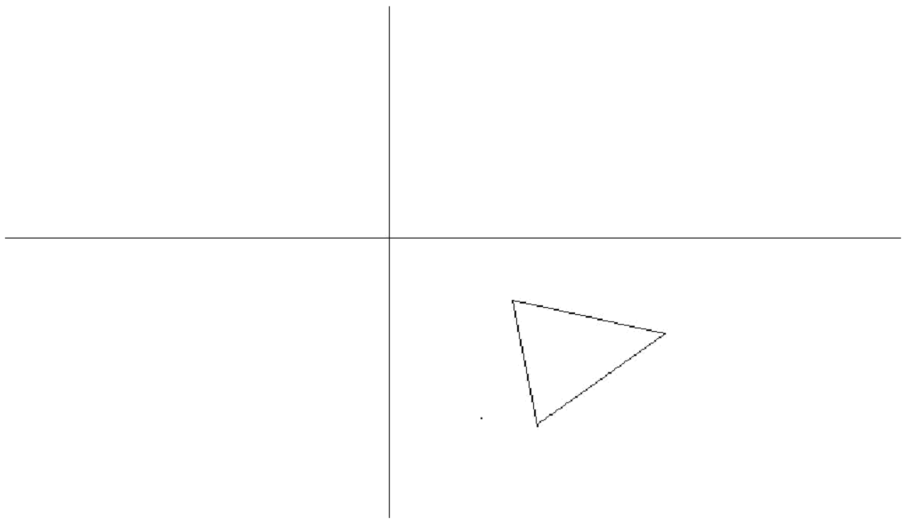
Program:

```
# include <iostream.h>
# include <conio.h>
# include <graphics.h>
# include <math.h>

char IncFlag;
int PolygonPoints[3][2] =
    {{10,100},{110,100},{110,200}};

void PolyLine()
{
    int iCnt;
    cleardevice();
    line(0,240,640,240);
    line(320,0,320,480);
    for (iCnt=0; iCnt<3; iCnt++)
    {
        line(PolygonPoints[iCnt][0],PolygonPoints[iCnt][1],
          PolygonPoints[(iCnt+1)%3][0],PolygonPoints[(iCnt+1)%3][1]);
    }
}
void Reflect()
{
    float Angle;
    int iCnt;
    int Tx,Ty;
    cout<<endl;
    for (iCnt=0; iCnt<3; iCnt++)
        PolygonPoints[iCnt][1] = (480 - PolygonPoints[iCnt][1]);
}

void main()
```

```
    int gDriver = DETECT, gMode;
    int iCnt;
    initgraph(&gDriver, &gMode, "C:\\TC\\BGI");
    for (iCnt=0; iCnt<3; iCnt++)
    {
        PolygonPoints[iCnt][0] += 320; PolygonPoints[iCnt][1] =
        240 - PolygonPoints[iCnt][1];
    }
    PolyLine();
    getch();
    Reflect();
    PolyLine();
    getch();
}
```

<u>**Output:-**</u>





Practical 10

AIM: To perform composite transformations of an object.

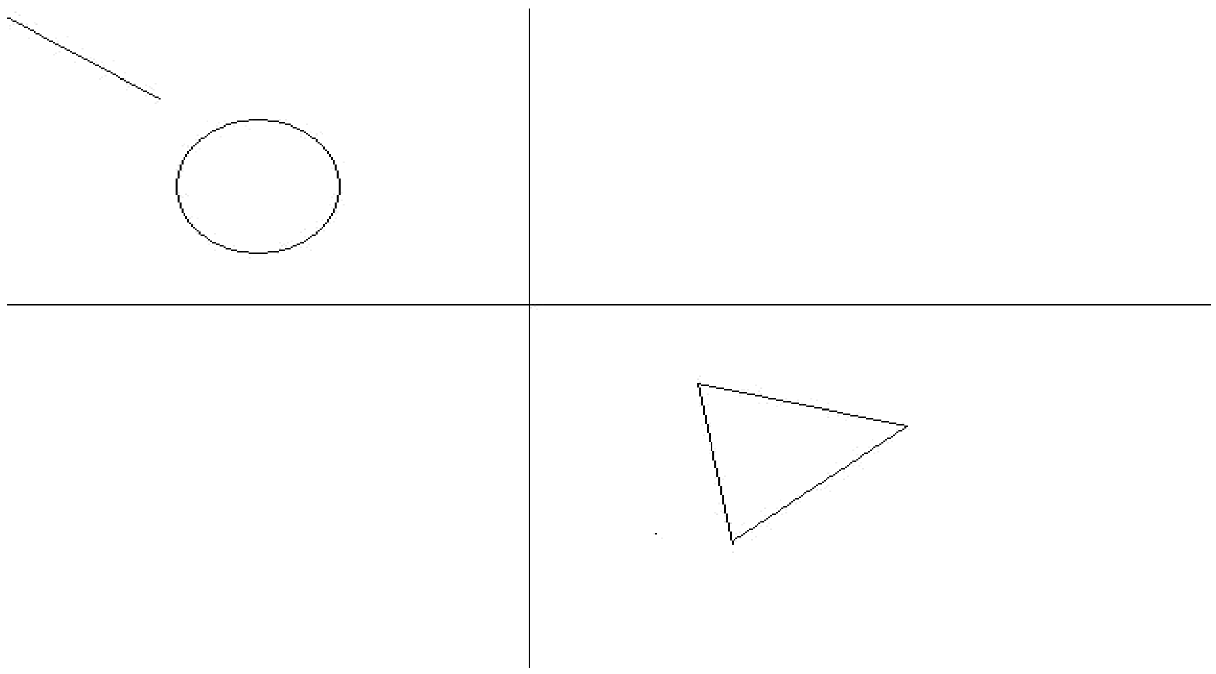
Objective:- In this program we combine more than one transformations.

Program:

```
# include <iostream.h>
# include <conio.h>
# include <graphics.h>
# include <math.h>

char IncFlag;
int PolygonPoints[3][2] =
    {{10,100},{110,100},{110,200}};

void PolyLine()
{
    int iCnt;
    cleardevice();
    line(0,240,640,240);
    line(320,0,320,480);
    for (iCnt=0; iCnt<3; iCnt++)
    {
        line(PolygonPoints[iCnt][0],PolygonPoints[iCnt][1],
          PolygonPoints[(iCnt+1)%3][0],PolygonPoints[(iCnt+1)%3][1]);
    }
}
void Reflect()
{
    float Angle;
    int iCnt;
    int Tx,Ty;
    cout<<endl;
    for (iCnt=0; iCnt<3; iCnt++)
        PolygonPoints[iCnt][1] = (480 - PolygonPoints[iCnt][1]);
}

void main()
```

```
int gDriver = DETECT, gMode; int
iCnt;
    initgraph(&gDriver, &gMode, "C:\\TC\\BGI");
    for (iCnt=0; iCnt<3; iCnt++)
    {
        PolygonPoints[iCnt][0] += 320; PolygonPoints[iCnt][1] =
        240 - PolygonPoints[iCnt][1];
    }
    PolyLine();
    getch();
    Reflect();
    PolyLine();
    line(0,0,50,50);
    circle(500,500,50);

    getch();
}
```

OUTPUT:



PRACTICAL 11

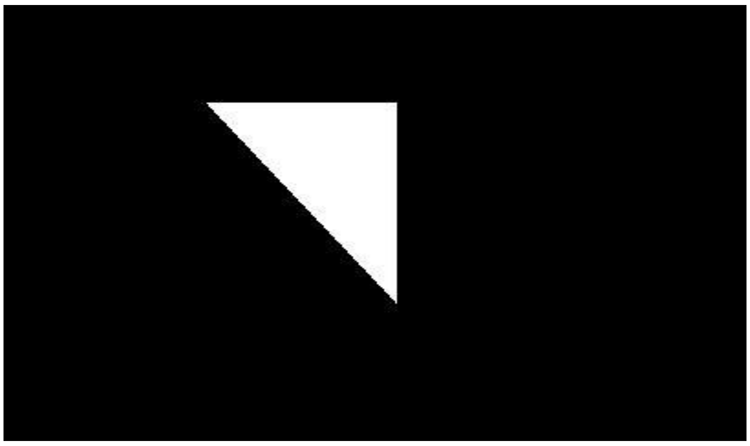
AIM:- Write a program for flood fill algorithm.

Objective:- To Fill the areas.

Program:-

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<graphics.h>

int main(void)
{
            int gd=DETECT,gm,errorcode;
            int ly;
            int poly[3];
            initgraph(&gd,&gm,””);
            poly[0]=300;
            poly[1]=300;
            poly[2]=300;
 for(i=0;i<10;i++)
{ setfillstyle(l,getmaxcolor());
          fillpoly(3,poly); getch();
}
closegraph();
          return 0;
}
```

Output :-



PRACTICAL 12

AIM- Write a program to fill for connected area of pixel by using boundary filling algorithm.

Objective:- To Fill the areas.

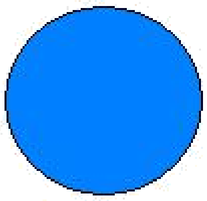
Program:-

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>

void bfill(int x,int y,int fill,int boun)

{
  int current;
  current=getpixel(x,
  y);

 if((current!=boun) && (current!=fill))

{

putpixel(x,y,fill);
bfill(x,y+1,fill,bou
n);           bfill(x,y-
1,fill,boun);
bfill(x+1,y,fill,bou
n);             bfill(x-
1,y,fill,boun);

}
}
void main()

{
  int gd=DETECT,gm;
   initgraph(&gd,&gm,"c:\tc:\bg
   i"); int x,y,r,fill,boun;

  cout<<"enter the values of x,y and
  r"; cin>>x>>y>>r;

  cout<<"enter the boundary and filling color
  values"; cin>>fill>>boun;
```

```
   setcolor(boun);
   circle(x,y,r);
  bfill(x,y,fill,boun);
 getch();

  closegraph();
}
```

<u>Output</u> :-

```
enter the radius to be
filled 30
enter the center coordinates of the circle to be
filled 250 200

enter the color code for the color to be
filled 6
enter the color code for the boundary
color 5
```



PRACTICAL 13

AIM- Perform the rotation of an object with certain angle about an arbitrary point

Program:-

```
#include"stdafx.h"
#include<math.h>
#include<ctype.h>
#include<iostream>
#define DegreesToRads
0.017453293f usingnamespace std;
//Define the Matrix3X3
type. typedefstruct
{
float
index[3][3]; }
Matrix3X3;
//Define the Matrix3X1
type. typedefstruct
{
float
index[3][1]; }
Matrix3X1;
Matrix3X3 createRotationCombo(float theta);
Matrix3X1 translate2DByMultiplication(Matrix3X1 start, float dx,
float dy); void rotate2D();
// Begin
main() void
main()
{
}
Matrix3X3 createRotationCombo(float theta)
{
Matrix3X3
temp;
Matrix3X1
result;
temp = createFixed3X3Matrix(0);
temp.index[0] [0] =
cos(DegreesToRads(theta)); temp.index[1]
[1] = cos(DegreesToRads(theta));
temp.index[2] [2] = 1;
temp.index[0] [1] = -
1*(sin(DegreesToRads(theta))); temp.index[1]
[0] = sin(DegreesToRads(theta)); temp.index[2]
[2] = 1;
result = multiplyMatrixNxM(temp,
start); return result;
}
Matrix 3X1 translate2DByMultiplication(Matrix3X1 start, float dx, float dy)
{
    Matrix 3X3 temp; Matrix3X1 result;
    //Zero out the matrix.
    temp = createFixed3X3Matrix(0); //setup the 3x3
    for multiplication temp.index[0][0] = 1;
    temp.index[1][1] = 1; temp.index[2][2] = 1;
    //put in the translation amount
    temp.index[0][2] = dx; temp.index[1][2] = dy;
    result = mulitplyMatrixNxM(temp,start); return result;

    void rotate2D
    {
    Matrix 3X1 A, B, C, temp;
    float vertx1, vertx2, vertx3, verty1, verty2, verty3, dx, dy, theta; cout<<"We
    will now translate and rotate a triangle in 2D. \n";
    cout<<"Please enter the coordinates, the value of the translation and the rotation.\n"
    cout<<"Enter the first vertex of the triangle's X value:\n"
    cin>>vertx1;
    cout<<"Enter the first vertex of the triangle's Y value:\n" cin>>verty1;
    cout<<"Enter the second vertex of the triangle's X value:\n; cin>>vertx2;
    cout<<"Enter the second vertex of the triangle's Y value:\n; cin>>verty2;
    cout<<"Enter the third vertex of the triangle's X value:\n; cin>>vertx3;
    cout<<"Enter the third vertex of the triangle's Y value:\n; cin>>verty3;
    cout<<"Enter the amount of dx units to translate horizontally:\n; cin>>dx;
    cout<<"Enter the amount of dy units to translate vertically:\n; cin>>dy;
    start.index[2] = 1; cout<<endl;
    cout<<"Enter the rotation angle in degrees:\n" cin>>theta;
    temp = rotate2D(A, B, C, theta);
    cout<<"The Y coordinate of the first vertex is <<verty1,,","<<A.index[1]<<"\n";
    cout<<"The X coordinate of the first vertex is <<vertx1,,","<<A.index[0]<<"\n";
    cout<<"The Y coordinate of the second vertex is <<verty2,,","<<B.index[1]<<"\n";
    cout<<"The X coordinate of the second vertex is <<vertx2,,","<<B.index[0]<<"\n";
    cout<<"The Y coordinate of the third vertex is <<verty3,,","<<C.index[1]<<"\n";
    cout<<"The X coordinate of the third vertex is <<vertx3,,","<<C.index[0]<<"\n";
    cout<<"The dy translation is <<dy,,","<<temp.index[1]<<"\n";
    cout<<"The dx translation is <<dx,,","<<temp.index[0]<<"\n";
    }
```