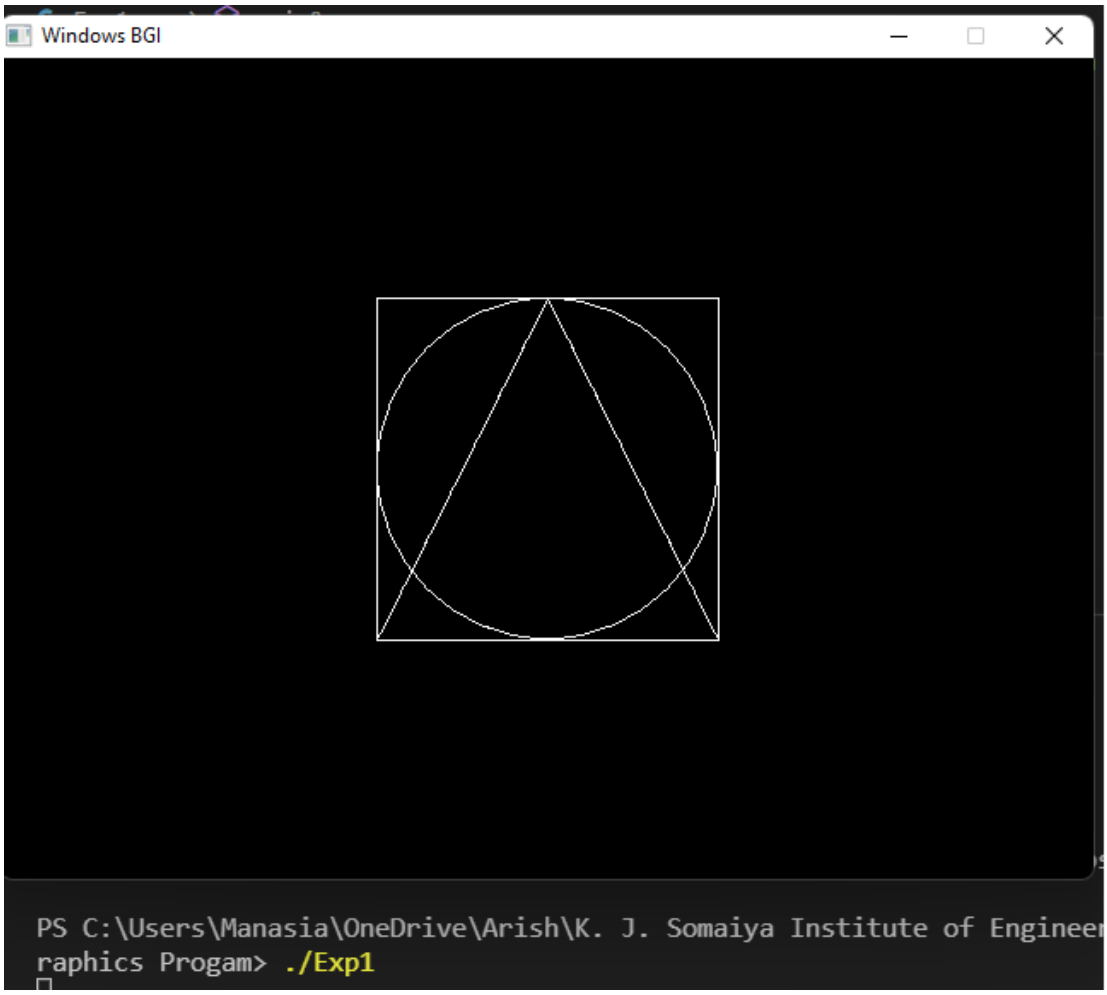
CG Codes:

Exp 1:

```
/*This program implements the sample program using graphics.h header file and
drawing various shapes*/
#include <iostream>
#include <graphics.h>
using namespace std;
int main()
{
    int gd=DETECT,gm;
    initgraph(&gd, &gm, (char*)"");
    rectangle (220,140,420,340);
    circle (320,240,100);
    line (320,140,220,340);
    line (320,140,420,340);
    getch();
            closegraph();
            return 0;
}
```



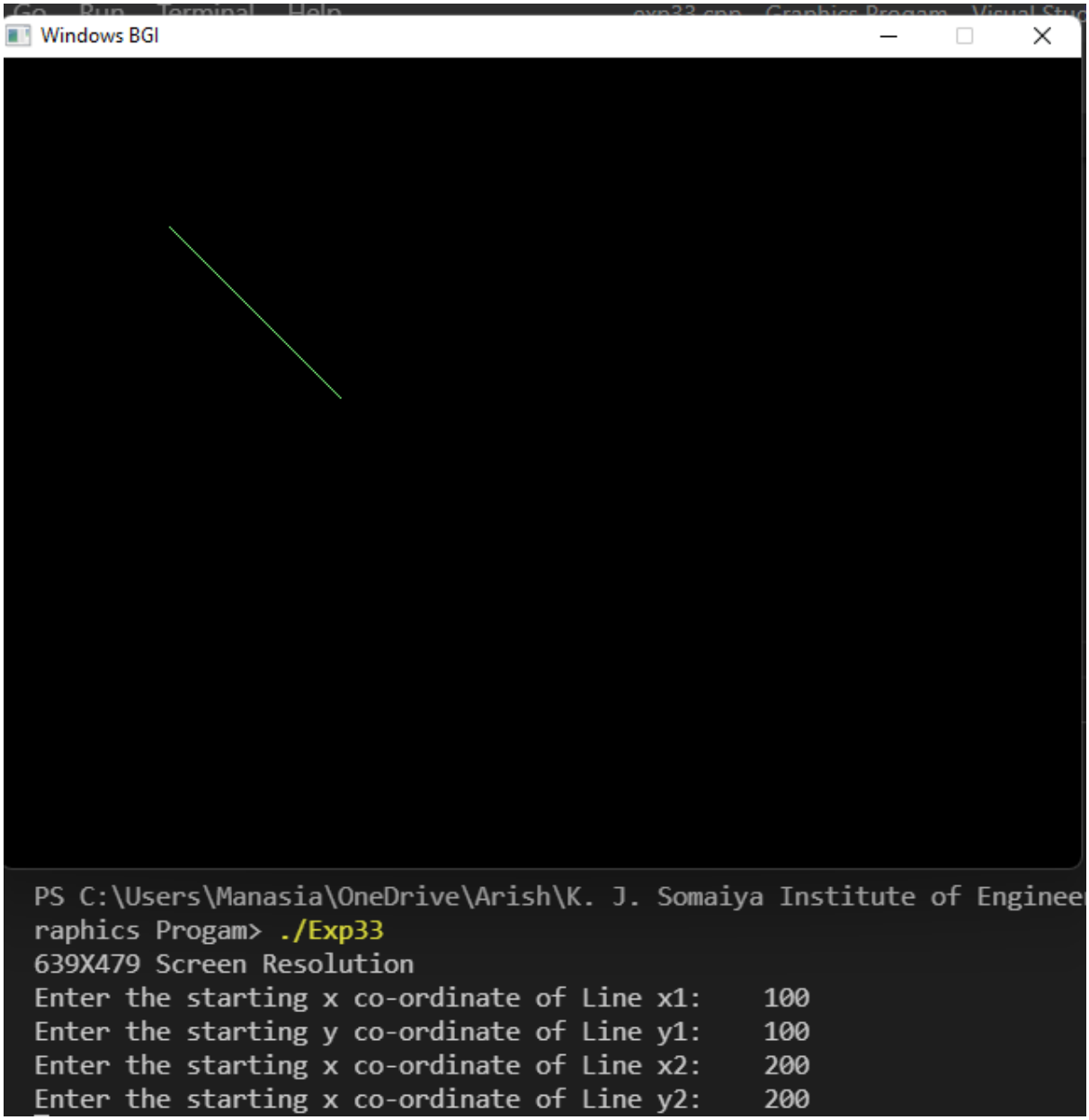
Exp 3:

```
/*This program implements the DDA Line drawing algorithm in C++*/

#include <iostream>

#include <graphics.h>
using namespace std;
int  main()
{
int gd,gm;
    gd=DETECT;
    initgraph(&gd,&gm,(char*)"");
    int x1,y1,x2,y2,dx,dy,m,i,x,y,a,b;
    a=getmaxx();
    b=getmaxy();
    cout<<a<<"X"<<b<<" Screen Resolution\n";
    cout<<"Enter the starting x co-ordinate of Line x1:\t";
    cin>>x1;
    cout<<"Enter the starting y co-ordinate of Line y1:\t";
    cin>>y1;
    cout<<"Enter the starting x co-ordinate of Line x2:\t";
    cin>>x2;
    cout<<"Enter the starting x co-ordinate of Line y2:\t";
    cin>>y2;
    dx=x2-x1;
    dy=y2-y1;
    m=dy/dx;
    x=x1;
    y=y1;
    line (x,y,x2,y2);
    putpixel(x1,y1,10);
    delay (50);
    if (dx>dy)
    {
        for (i=x1;i<=x2;i++)
        {
            x1++;
            y1=y1+m;
            putpixel(x1,y1,10);
            delay(50);
        }
    }
    else
    {
        for (i=y1;i<=y2;i++)
        {
            y1++;
            x1=x1+(1/m);
```

```
            putpixel(x1,y1,10);
            delay(50);
        }
    }
    getch();
    closegraph();
return 0;
}
```



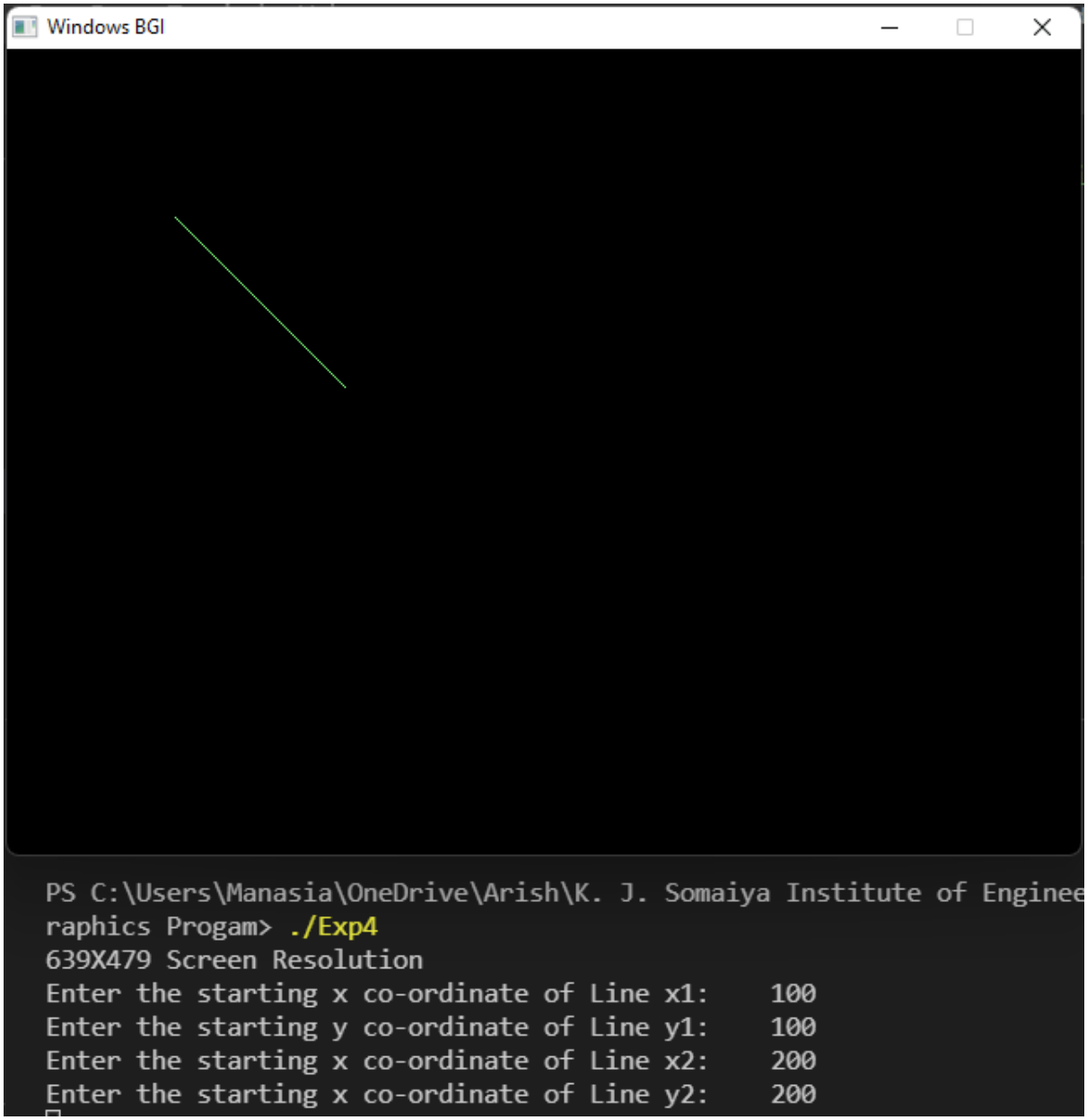
Exp 4:

```
/*This program implements the Bresenham's Line drawing algorithm in C++*/

#include <iostream>

#include <graphics.h>
using namespace std;
int main()
{
    int gd,gm;
    gd=DETECT;
    initgraph(&gd,&gm,(char*)"");
    int x1,y1,x2,y2,dx,dy,i,x,y,a,b,p=0;
    a=getmaxx();
    b=getmaxy();
    cout<<a<<"X"<<b<<" Screen Resolution\n";
    cout<<"Enter the starting x co-ordinate of Line x1:\t";
    cin>>x1;
    cout<<"Enter the starting y co-ordinate of Line y1:\t";
    cin>>y1;
    cout<<"Enter the starting x co-ordinate of Line x2:\t";
    cin>>x2;
    cout<<"Enter the starting x co-ordinate of Line y2:\t";
    cin>>y2;
    dx=x2-x1;
    dy=y2-y1;
    x=x1;
    y=y1;
    line (x,y,x2,y2);
    putpixel(x1,y1,10);
    delay (50);
    p=(2*(dy))-(2*(dx));
    i=0;
    do
    {
        if (p<0)
        {
            x1++;
            putpixel(x1,y1,10);
            delay(50);
            p=p+2*(dy);
        }
        else
        {
            x1++;
            y1++;
            putpixel(x1,y1,10);
            delay(50);
```

```
            p=p+((2*(dy))-(2*(dx)));
        }
        i++;
    } while (i<=dx);
    getch();
    closegraph();
return 0;
}
```



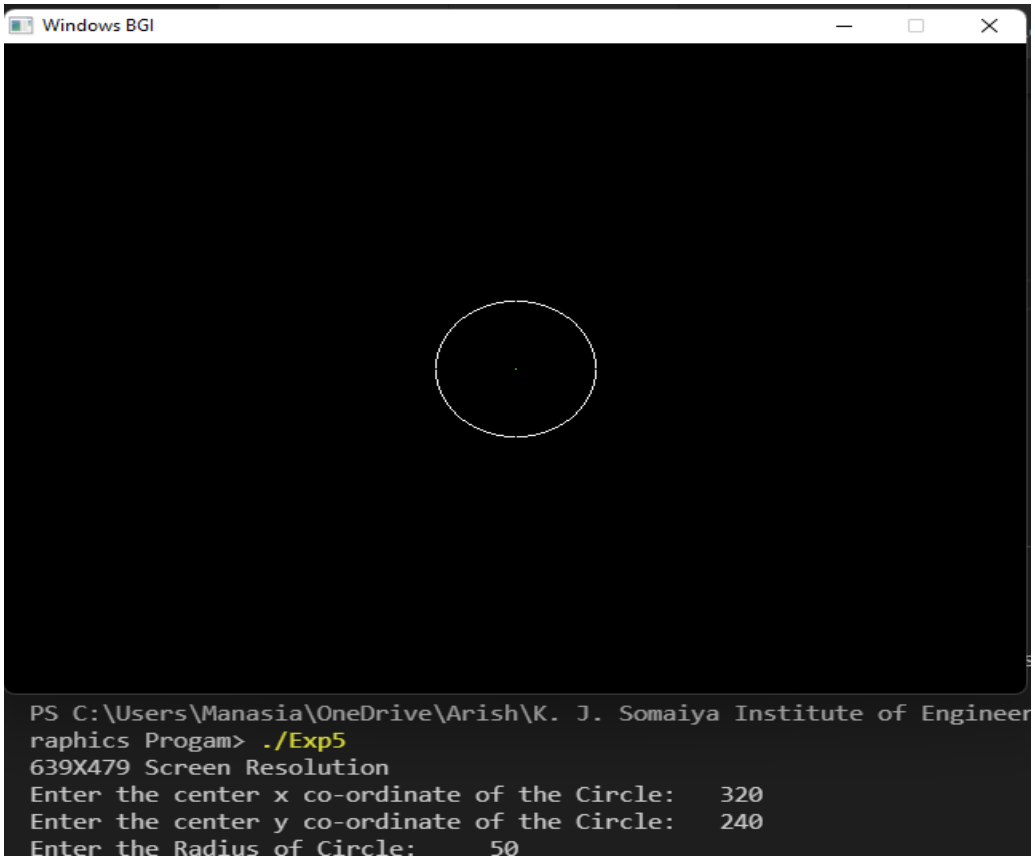
Exp 5:

```
/*This program implements the Mid-point Circle drawing algorithm in C++*/

#include <iostream>

#include <graphics.h>
using namespace std;
void display(int xc, int yc, float x, float y)
{
    putpixel (xc+x,yc+y,15);
    delay (10);
    putpixel (xc+y,yc+x,15);
    delay (10);
    putpixel (xc+x,yc-y,15);
    delay (10);
    putpixel (xc+y,yc-x,15);
    delay (10);
    putpixel (xc-x,yc+y,15);
    delay (10);
    putpixel (xc-y,yc+x,15);
    delay (10);
    putpixel (xc-x,yc-y,15);
    delay (10);
    putpixel (xc-y,yc-x,15);
    delay (10);
}
int  main()
{
    int gd,gm;
    gd=DETECT;
    initgraph(&gd,&gm,(char*)"");
    int r,xc,yc,a,b;
    float x,y,d;
    a=getmaxx();
    b=getmaxy();
    cout<<a<<"X"<<b<<" Screen Resolution\n";
    cout<<"Enter the center x co-ordinate of the Circle:\t";
    cin>>xc;
    cout<<"Enter the center y co-ordinate of the Circle:\t";
    cin>>yc;
    cout<<"Enter the Radius of Circle:\t";
    cin>>r;
    //circle (xc,yc,r);
    x=0;
    y=r;
    putpixel (xc,yc,10);
    delay (50);
    d=(1.25)-r;
```

```
    while (x<=y)
    {
        if (d<0)
        {
            x++;
            d=d+(2*x)+1;
        }
        else
        {
            x++;
            y--;
            d=d+(2*(x-y))+1;
        }
        display (xc,yc,x,y);
    }
    getch();
    closegraph();
return 0;
}
```



Exp 6:

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

void flood_fill(int x, int y, int ncolor, int ocolor)
{
    if (getpixel(x, y) == ocolor) {
    putpixel(x, y, ncolor);
    flood_fill(x + 1, y, ncolor, ocolor);
    flood_fill(x + 1, y - 1, ncolor, ocolor);
    flood_fill(x + 1, y + 1, ncolor, ocolor);
    flood_fill(x, y - 1, ncolor, ocolor);
    flood_fill(x, y + 1, ncolor, ocolor);
    flood_fill(x - 1, y, ncolor, ocolor);
    flood_fill(x - 1, y - 1, ncolor, ocolor);
    flood_fill(x - 1, y + 1, ncolor, ocolor);
    }
}


 main()
{
    int x, y, ncolor, ocolor;

    printf("Enter the seed point (x,y): ");
    scanf("%d%d", &x, &y);
    printf("Enter old color : ");
    scanf("%d", &ocolor);
    printf("Enter new color : ");
    scanf("%d", &ncolor);

    int gd = DETECT, gm = DETECT;
    initgraph(&gd,&gm,(char*)"");
    cleardevice();

    /*

       Draw some figures
       to create closed shapes
       you can use any biut-in functions.
       Everyone, please make different figures.

     */
    circle(300, 200, 50);

    flood_fill(x, y, ncolor, ocolor);
```

```
    getch();
}
```

Exp 7:

```
/*This program implements the program on 2-Dimentional Transformation in C++*/
#include <iostream>
#include <graphics.h>
#include <math.h>
using namespace std;
int main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,(char*)"");
    int tx,ty;
    int x,y,n,i,a[10][3],choice,t[3][3],k,j,vrf;
    float angle,ans[10][3],sx,sy;
    cleardevice();
    x=getmaxx();
    y=getmaxy();
    cout<<x<<"X"<<y<<" Screen Resolution\n";
    cout<<"Enter the number of vertices less than 10:\t";
    cin>>n;
    for (i=0;i<n;i++)
    {
        cout<<"Enter the x co-ordinate of Vertex "<<(i+1)<<":\t";
        cin>>a[i][0];
        cout<<"Enter the y co-ordinate of Vertex "<<(i+1)<<":\t";
        cin>>a[i][1];
        a[i][2]=1;
    }
    cleardevice();
    for (i=0;i<n-1;i++)
        line (a[i][0],a[i][1],a[i+1][0],a[i+1][1]);
    line (a[n-1][0],a[n-1][1],a[0][0],a[0][1]);
    getch();
    do
    {
        cleardevice();
        cout<<"Enter your choice\n";
        cout<<"1.   Translation\n";
        cout<<"2.   Rotation\n";
        cout<<"3.   Scaling\n";
        cout<<"4.   Exit\n";
```

```
        cin>>choice;
        switch (choice)
        {
            case 1:
                cout<<"Enter the X co-ordinate Translation Value tx:\t";
                cin>>tx;
                cout<<"Enter the Y co-ordinate Translation Value ty:\t";
                cin>>ty;
                t[0][0]=1;
                t[0][1]=0;
                t[0][2]=0;
                t[1][0]=0;
                t[1][1]=1;
                t[1][2]=0;
                t[2][0]=tx;
                t[2][1]=ty;
                t[2][2]=1;
                for (i=0;i<n;i++)
                    for (j=0;j<3;j++)
                        ans[i][j]=(a[i][0]*t[0][j])+(a[i][1]*t[1][j])+(a[i][2]
*t[2][j]);
                /*Original image*/
                cleardevice();
                for (i=0;i<n-1;i++)
                    line (a[i][0],a[i][1],a[i+1][0],a[i+1][1]);
                line (a[n-1][0],a[n-1][1],a[0][0],a[0][1]);
                getch();
                /*Translated Image*/
                for (i=0;i<n-1;i++)
                    line (ans[i][0],ans[i][1],ans[i+1][0],ans[i+1][1]);
                line (ans[n-1][0],ans[n-1][1],ans[0][0],ans[0][1]);
                getch();
                break;
            case 2:
                cout<<"Enter the Rotation Angle:\t";
                cin>>angle;
                angle=angle*(3.14/180);
                cout<<"Angle Value is:\t"<<angle<<"\n";
                cout<<"The Value of cos("<<angle<<") is:\t"<<cos(angle)<<"\n";
                cout<<"The Value of sin("<<angle<<") is:\t"<<sin(angle)<<"\n";
                cout<<"Enter the reference Vertex:\t";
                cin>>vrf;
                for (i=0;i<n;i++)
                {
                    ans[i][0]=((a[i][0]-a[vrf-1][0])*cos(angle))-((a[i][1]-
a[vrf-1][1])*sin(angle))+a[vrf-1][0];
                    ans[i][1]=((a[i][0]-a[vrf-1][0])*sin(angle))-((a[i][1]-
a[vrf-1][1])*cos(angle))+a[vrf-1][1];
```

```
            }
            /*Original image*/
            cleardevice();
            for (i=0;i<n-1;i++)
                line (a[i][0],a[i][1],a[i+1][0],a[i+1][1]);
            line (a[n-1][0],a[n-1][1],a[0][0],a[0][1]);
            getch();
            /*Translated Image*/
            for (i=0;i<n-1;i++)
                line (ans[i][0],ans[i][1],ans[i+1][0],ans[i+1][1]);
            line (ans[n-1][0],ans[n-1][1],ans[0][0],ans[0][1]);
            getch();
            break;
        case 3:
            cout<<"Enter the X co-ordinate Scaling Value sx:\t";
            cin>>sx;
            cout<<"Enter the Y co-ordinate Scaling Value sy:\t";
            cin>>sy;
            cout<<"Enter the reference Vertex:\t";
            cin>>vrf;
            for (i=0;i<n;i++)
            {
                ans[i][0]=(a[i][0]-a[vrf-1][0])*sx+a[vrf-1][0];
                ans[i][1]=(a[i][1]-a[vrf-1][1])*sy+a[vrf-1][1];
            }
            /*Original image*/
            cleardevice();
            for (i=0;i<n-1;i++)
                line (a[i][0],a[i][1],a[i+1][0],a[i+1][1]);
            line (a[n-1][0],a[n-1][1],a[0][0],a[0][1]);
            getch();
            /*Translated Image*/
            for (i=0;i<n-1;i++)
                line (ans[i][0],ans[i][1],ans[i+1][0],ans[i+1][1]);
            line (ans[n-1][0],ans[n-1][1],ans[0][0],ans[0][1]);
            getch();
            break;
        case 4:
            cout<<"Thank you";
            break;

        default:
            cout<<"Invalid Choice";
            break;
    }
}while (choice!=4);
getch();
closegraph();
```

```
return 0;
}
```