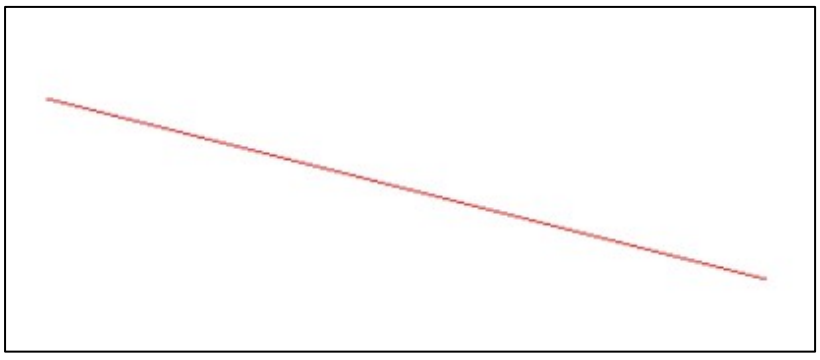
Practical No: 1

Objective: **Implementation of line generation DDA algorithm.**

Source Code:-

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main(){
   int gdriver = DETECT, gmode, length, i;
   float x, y, dx, dy;
   int x0, y0, x1, y1;
initgraph(&gdriver,&gmode,"c:\\turboc3\\bgi");
setbkcolor(WHITE);
   x0 = 100, y0 = 200, x1 = 500, y1 = 300;
   dx = x1-x0;
dy = y1-y0;
   x = x0;
   y = y0;
   if(dx>=dy)
      length = dx;
   else
      length = dy;
   dx = dx/length;
dy = dy/length;
i=1;
   while(i<=length){
putpixel(x, y, RED);
      x = x+dx;
      y = y+dy;
i++;}
getch();
closegraph();
}
```

OUTPUT:-



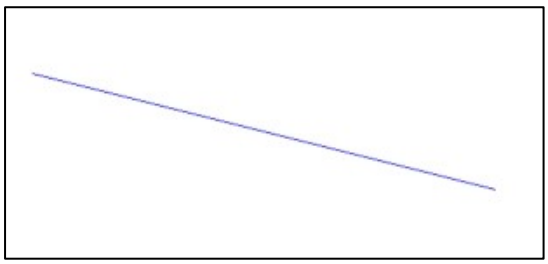
Practical No: 2

Objective: **Implementation of line generation Bresenham's algorithm.**

Source Code:-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void drawline(int x0, int y0, int x1, int y1){
        int dx, dy, p, x, y;
        dx=x1-x0;
        dy=y1-y0;
        x=x0;
        y=y0;
        p=2*dy-dx;
        while(x<x1){
                if(p>=0){
                        putpixel(x,y,BLUE);
                        y=y+1;
                        p=p+2*dy-2*dx;    }
                else{
                        putpixel(x,y,BLUE);
                        p=p+2*dy;    }
                x=x+1;    }
}
int main() {
        int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
        initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
        setbkcolor(WHITE);
        x0=100,y0=200,x1=500,y1=300;
        drawline(x0, y0, x1, y1);
        getch();
        return 0;
}
```

Output:-



Practical No: 3

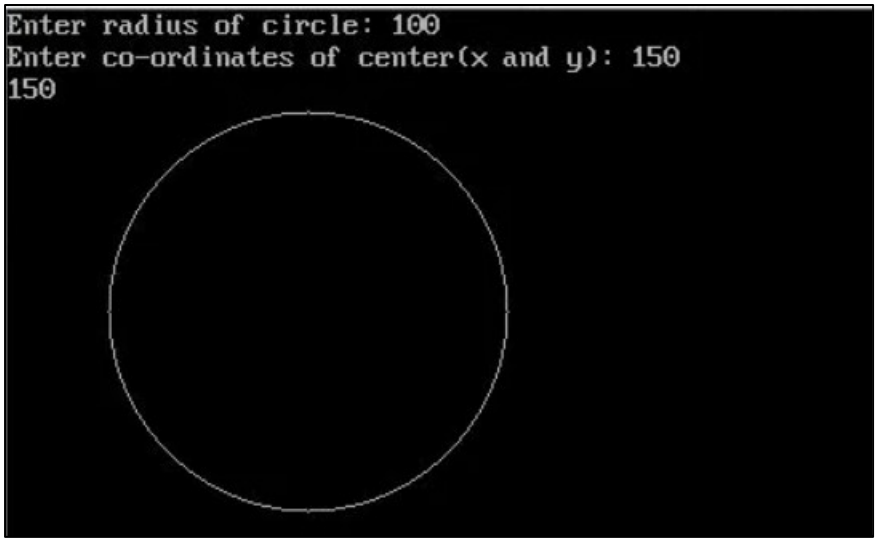
Objective: **Implementation of circle generation using mid-point method.**

Source Code:-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void drawcircle(int x0, int y0, int radius)
{
   int x = radius;
   int y = 0;
   int err = 0;

   while (x >= y)
   {
         putpixel(x0 + x, y0 + y, 7);
         putpixel(x0 + y, y0 + x, 7);
         putpixel(x0 - y, y0 + x, 7);
         putpixel(x0 - x, y0 + y, 7);
         putpixel(x0 - x, y0 - y, 7);
         putpixel(x0 - y, y0 - x, 7);
         putpixel(x0 + y, y0 - x, 7);
         putpixel(x0 + x, y0 - y, 7);

         if (err <= 0)
         {
            y += 1;
            err += 2*y + 1;
         }

         if (err > 0)
         {
            x -= 1;
            err -= 2*x + 1;
         }
   }
}

int main()
{
int gdriver=DETECT, gmode, error, x, y, r;
initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

printf("Enter radius of circle: ");
```

```
scanf("%d", &r);
printf("Enter co-ordinates of center(x and y): ");
scanf("%d%d", &x, &y);
drawcircle(x, y, r);
getch();
return 0;
}
```

Output:-



Practical No: 4

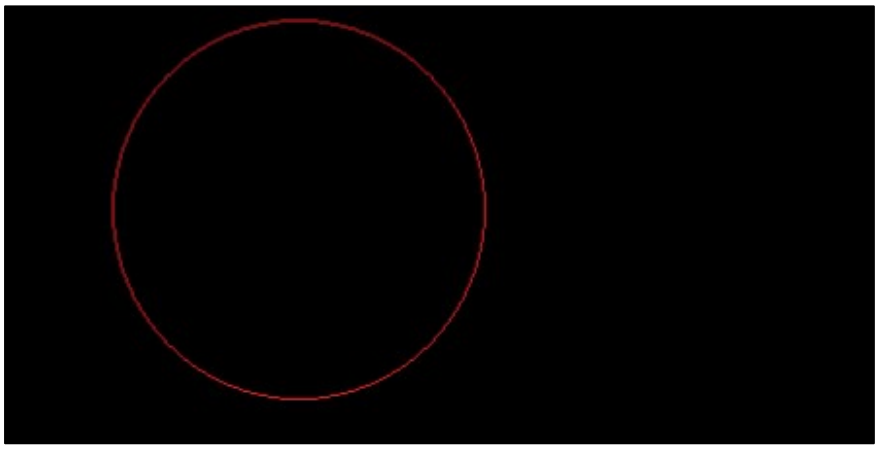
Objective: **Implementation of circle generation using Bresenham's algorithm.**

Source Code:-

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

void drawCircle(int xc, int yc, int x, int y)
{
putpixel(xc+x, yc+y, RED);
putpixel(xc-x, yc+y, RED);
putpixel(xc+x, yc-y, RED);
putpixel(xc-x, yc-y, RED);
putpixel(xc+y, yc+x, RED);
putpixel(xc-y, yc+x, RED);
putpixel(xc+y, yc-x, RED);
putpixel(xc-y, yc-x, RED);
}
void circleBres(int xc, int yc, int r)
{
   int x = 0, y = r;
   int d = 3 - 2 * r;
drawCircle(xc, yc, x, y);
   while (y >= x)
   {
      x++;
      if (d > 0)
      {
         y--;
         d = d + 4 * (x - y) + 10;
      }
      else
         d = d + 4 * x + 6;
drawCircle(xc, yc, x, y);
   }
}

int main()
{
   int x0 = 50, y0 = 50, r = 30;
   int gdriver = DETECT, gmode;
initgraph(&gdriver, &gmode, "");
circleBres(x0, y0, r);
```

```
getch();
   return 0;
}
```

Output:-



Practical No: 5

Objective: **Implementation of polygon filling using boundary-fill algorithm.**

Source Code:-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void b_fill(int x, int y, int f_col, int b_col)
{
if(getpixel(x,y)!=b_col && getpixel(x,y)!=f_col)
        {
        putpixel(x, y, f_col);
        b_fill(x, y+1, f_col, b_col);
        b_fill(x, y-1, f_col, b_col);
        b_fill(x+1, y, f_col, b_col);
        b_fill(x-1, y, f_col, b_col);
        b_fill(x-1, y+1, f_col, b_col);
        b_fill(x-1, y-1, f_col, b_col);
        b_fill(x+1, y+1, f_col, b_col);
        b_fill(x+1, y-1, f_col, b_col);
        }
}
int main()
{
        int gdriver=DETECT, gmode;
        initgraph(&gdriver, &gmode, "");
        setbkcolor(WHITE);
        setcolor(BLUE);
        cleardevice();
        rectangle(10, 10, 100, 100);
        b_fill(50, 50, GREEN, BLUE);
        getch();
        return 0;
}
```

Output:-

Practical No: 6

Objective: **Implementation of polygon filling using flood-fill algorithm.**

Source Code:-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void f_fill(int x, int y, int f_col, int old_col) {
    if(getpixel(x,y)==old_col) {
        putpixel(x, y, f_col);
        f_fill(x, y+1, f_col, old_col);
        f_fill(x, y-1, f_col, old_col);
        f_fill(x+1, y, f_col, old_col);
        f_fill(x-1, y, f_col, old_col);
        f_fill(x-1, y+1, f_col, old_col);
        f_fill(x-1, y-1, f_col, old_col);
        f_fill(x+1, y+1, f_col, old_col);
        f_fill(x+1, y-1, f_col, old_col);
        }
        }
int main() {
        int gdriver=DETECT, gmode;
        initgraph(&gdriver, &gmode, "");
        setbkcolor(WHITE);
        setcolor(BLUE);
        cleardevice();
        rectangle(10, 10, 100, 100);
        f_fill(50, 50, GREEN, WHITE);
        getch();
        return 0;
}
```

Output:-

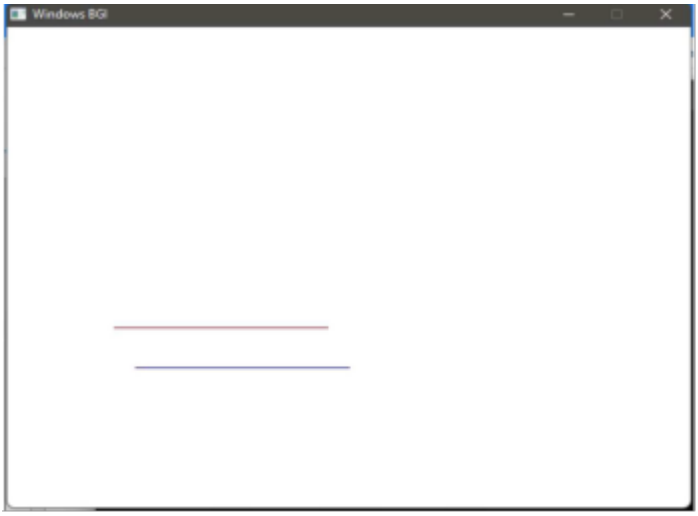
Practical No: 7

Objective: **Implementation of 2D transformation: Translation.**

Source Code:-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main()
{
        int gd=DETECT, gm;
        int x1=100, y1=300, x2=300, y2=300, tx=20, ty=40;
        initgraph(&gd,&gm," ");
        setbkcolor(WHITE);
        cleardevice();
        setcolor(RED);
        line(x1+tx, y1+ty, x2+tx, y2+ty);
        getch();
        return 0;
}
```

Output:-



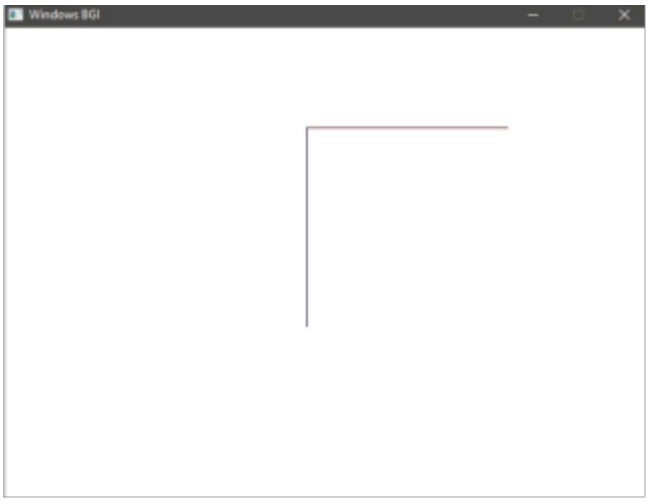
Practical No: 8

Objective: **Implementation of 2D transformation: Rotation.**

Source Code:-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
int main()
{
        int gd=DETECT, gm;
        int x1=300, y1=100, x2=500, y2=100, x3, y3;
        double degree=90, radian;
        initgraph(&gd,&gm," ");
        setbkcolor(WHITE);
        cleardevice();
        setcolor(RED);
        line(x1, y1, x2, y2);
        radian = degree*0.01745;
        x3 = (int)(x1+(x2-x1)*cos(radian)-(y2-y1)*sin(radian)));
        y3 = (int)(y1+(x2-x1)*sin(radian)+(y2-y1)*cos(radian)));
        setcolor(BLUE);
        line(x1,y1,x3,y3);
        getch();
        return 0;
}
```

Output:-



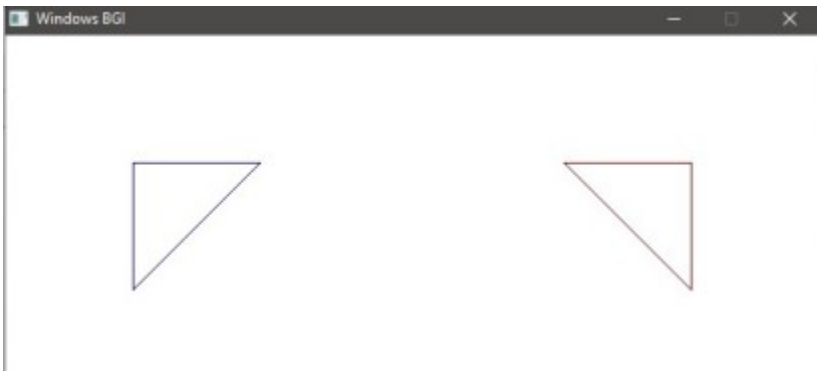
Practical No: 9

Objective: **Implementation of 2D transformation: Mirror Reflection.**

Source Code:-

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
int main()
{
        int gm, gd = DETECT, ax;
        int x1 = 100, x2 = 100, x3 = 200;
        int y1 = 100, y2 = 200, y3 = 100;
        initgraph(&gd, &gm, "");
        setbkcolor(WHITE);
        cleardevice();
        line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());
        line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);
        setcolor(BLUE);
        line(x1, y1, x2, y2);
        line(x2, y2, x3, y3);
        line(x3, y3, x1, y1);
        setcolor(RED);
        line(getmaxx() - x1, y1, getmaxx() - x2, y2);
        line(getmaxx() - x2, y2, getmaxx() - x3, y3);
        line(getmaxx() - x3, y3, getmaxx() - x1, y1);
        getch();
        return 0;
}
```

Output:-



Practical No: 10

Objective: **Implementation of ellipse generation using mid-point method.**

Source Code:-

```
#include<conio.h>
#include<stdio.h>
#include<graphics.h>
int main(){
int gd=DETECT,gm;
float x,y,xc=100,yc=100,rx=60,ry=40,pk,pk1;
initgraph(&gd,&gm," ");
setbkcolor(WHITE);
cleardevice();
x=0;
y=ry;
pk=(ry*ry)-(rx*rx*ry)+((rx*rx)/4);
while((2*x*ry*ry)<(2*y*rx*rx)){
if(pk<=0){
x=x+1;
pk1=pk+(2*ry*ry*x)+(ry*ry);
}
else{
x=x+1;
y=y-1;
pk1=pk+(2*ry*ry*x)-(2*rx*rx*y)+(ry*ry);
}
pk=pk1;
putpixel(xc+x,yc+y,2);
putpixel(xc-x,yc+y,2);
putpixel(xc+x,yc-y,2);
putpixel(xc-x,yc-y,2);
}
pk=((x+0.5)*(x+0.5)*ry*ry)+((y-1)*(y-1)*rx*rx)-(rx*rx*ry*ry);
while(y>0){
if(pk>0){
y=y-1;
pk1=pk-(2*rx*rx*y)+(rx*rx);
}
else{
x=x+1;
y=y-1;
pk1=pk+(2*ry*ry*x)-(2*rx*rx*y)+(rx*rx);
}
pk=pk1;
putpixel(xc+x,yc+y,2);
putpixel(xc-x,yc+y,2);
putpixel(xc+x,yc-y,2);
putpixel(xc-x,yc-y,2);
}
getch();
```

}

Output:-

Practical No: 11

Objective: **Implementation of line clipping using Cohen-Sutherland algorithm.**

Source Code:-

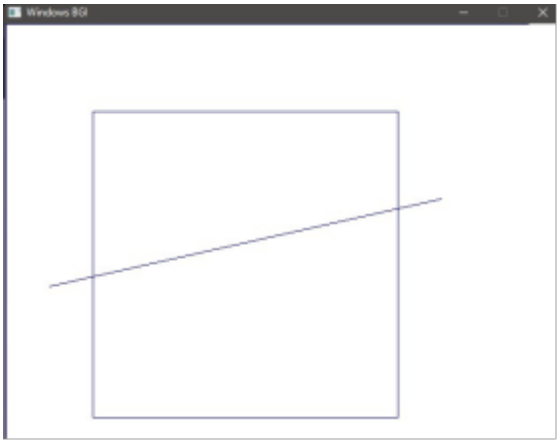
```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
int main(){
int rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4];
int W_xmax=450,W_ymax=450,W_xmin=100,W_ymin=100,flag=0;
float slope;
int x=500,y=200,x1=50,y1=300,i, xc,yc;
int gr=DETECT,gm;
initgraph(&gr,&gm," ");
setbkcolor(WHITE);
setcolor(BLUE);
cleardevice();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(x,y,x1,y1);
line(0,0,600,0);
line(0,0,0,600);
if(y>W_ymax) {
rcode_begin[0]=1;
flag=1 ;
}
if(y<W_ymin) {
rcode_begin[1]=1;
flag=1;
}
if(x>W_xmax) {
rcode_begin[2]=1;
flag=1;
}
if(x<W_xmin) {
rcode_begin[3]=1;
flag=1;
}i
f(y1>W_ymax) {
rcode_end[0]=1;
flag=1;
}i
f(y1<W_ymin) {
rcode_end[1]=1;
flag=1;
}i
f(x1>W_xmax) {
rcode_end[2]=1;
flag=1;
}i
```

```
f(x1<W_xmin) {
rcode_end[3]=1;
flag=1;
}i
f(flag==0) {
printf("No need of clipping as it is already in window");
}
flag=1;
for(i=0;i<4;i++){
region_code[i]= rcode_begin[i] && rcode_end[i] ;
if(region_code[i]==1)
flag=0;
}
if(flag==0){
printf("\n Line is completely outside the window");
}
else{
slope=(float)(y1-y)/(x1-x);
if(rcode_begin[2]==0 && rcode_begin[3]==1) {
y=y+(float) (W_xmin-x)*slope ;
x=W_xmin;
}
if(rcode_begin[2]==1 && rcode_begin[3]==0) {
y=y+(float) (W_xmax-x)*slope ;
x=W_xmax;
}i
f(rcode_begin[0]==1 && rcode_begin[1]==0) {
x=x+(float) (W_ymax-y)/slope ;
y=W_ymax;
}i
f(rcode_begin[0]==0 && rcode_begin[1]==1) {
x=x+(float) (W_ymin-y)/slope ;
y=W_ymin;
}i
f(rcode_end[2]==0 && rcode_end[3]==1) {
y1=y1+(float) (W_xmin-x1)*slope ;
x1=W_xmin;
}i
f(rcode_end[2]==1 && rcode_end[3]==0) {
y1=y1+(float) (W_xmax-x1)*slope ;
x1=W_xmax;
}i
f(rcode_end[0]==1 && rcode_end[1]==0) {
x1=x1+(float) (W_ymax-y1)/slope ;
y1=W_ymax;
}
if(rcode_end[0]==0 && rcode_end[1]==1) {
x1=x1+(float) (W_ymin-y1)/slope ;
y1=W_ymin;
}
}
delay(2000);
clearviewport();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(0,0,600,0);
```

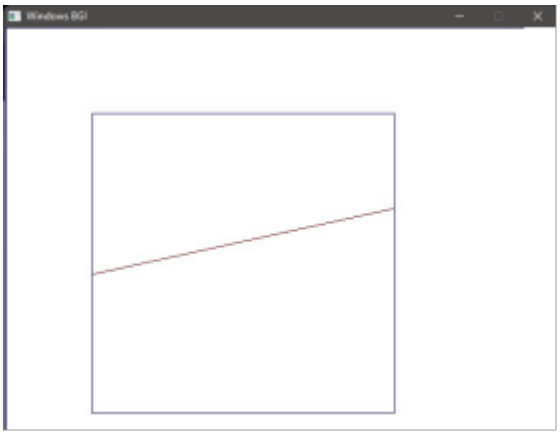
```
line(0,0,0,600);
setcolor(RED);
line(x,y,x1,y1);
getch();
closegraph();
}
```

Output:-

Before Clipping:



After Clipping:



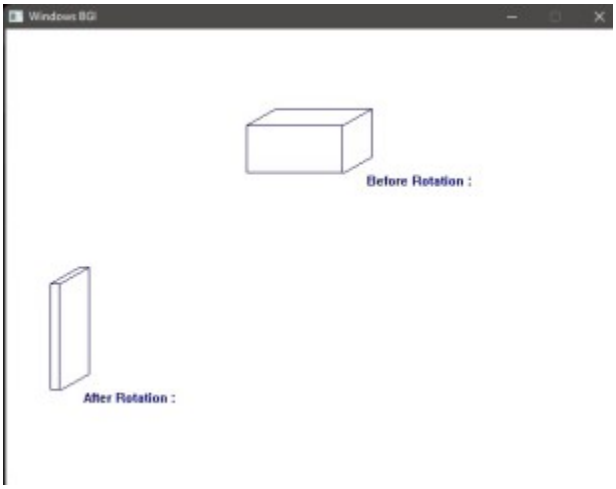
Practical No: 12

Objective: **Implementation of 3d geometric transformations: Rotation.**

Source Code:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int main() {
int gd=DETECT, gm;
initgraph (&gd, &gm, "");
int x1=250,y1=100, x2=350, y2=150, theta=45;
int a1, b1, a2, b2;
setbkcolor(WHITE);
cleardevice();
setcolor(BLUE);
a1=x1*cos(theta)-y1*sin(theta);
b1=x1*sin(theta)+y1*cos(theta);
a2=x2*cos(theta)-y2*sin(theta);
b2=x2*sin(theta)+y2*cos(theta);
bar3d(x1, y1, x2, y2, 30, 1) ;
outtextxy(x2+25, y2, "Before Rotation :");
bar3d(a1, b1, a2,b2, 30,1);
outtextxy(a2+25,b2,"After Rotation :");
getch();
closegraph();
return 0;
}
```

Output:-



Practical No: 13

Objective: **Implementation of curve generation using B-Spline curves.**

Source Code:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int main(){
int x[4],y[4],i;
double puty,putx,t;
int gd=DETECT,gm;
initgraph(&gd,&gm,"");
setbkcolor(WHITE);
cleardevice();
for ( i = 0; i < 4; i++) {
printf("Enter x and y coordinated of point %d: ",i+1);
scanf("%d%d",&x[i],&y[i]);
putpixel(x[i],y[i],3); }
for(t=0.0;t<=1.0;t=t+0.001){
putx=pow(1-t,3)*x[0]+ 3*t*pow(1-t,2)*x[1]+ 3*t*t*pow(1-t,1)*x[2]+ pow(t,3)*x[3];
puty=pow(1-t,3)*y[0]+ 3*t*pow(1-t,2)*y[1]+ 3*t*t*pow(1-t,1)*y[2]+ pow(t,3)*y[3];
putpixel(putx,puty,BLUE); }
getch();
closegraph();
return 0;
}
```

Output:-