Delhi University

B.Sc. (Computer Science)

Computer Graphics Practical File (19020570018)

Submitted By:- K A Shankar Narayan(20191417)
Submitted To:- Ms. Sheetal

1. Write a program to implement Bresenhams line drawingalgorithm.

```
#include<stdio.h>
#include<graphics.h>
void midpointline(int x0,int y0,int x1,int y1,int value)
{
double dy=y1-y0;
double dx=x1-x0;
int  d=2*dy-dx;
int incrE=2*dy;
int increNE=2*(dy-dx);
int x=x0;
int y=y0;
putpixel(x,y,value);
while(x<x1)
{                       //choose E
if(d<0)
{
d+=incrE;
x++;
}
else
{
d+=increNE;
x++;
y++;
}
putpixel(x,y,value);
}
}
main()
```

```
{
int gd=DETECT,gm;
int i,j,k;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
midpointline(100,100,200,200,3);
getch();
closegraph();
return 0;
}
```

2. Write a program to implement mid-point circle drawing algorithm.

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

  void circlepoints(int x1,int y1,int x, int y,int val)
  {
        putpixel(x1+x,y1+y,val);
        putpixel(x1+y,y1+x,val);
        putpixel(x1+y,y1-x,val);
        putpixel(x1+x,y1-y,val);
        putpixel(x1-x,y1-y,val);
        putpixel(x1-y,y1-x,val);
        putpixel(x1-x,y1+y,val);
        putpixel(x1-y,y1+x,val);
  }

  void midpointcircle(int x1,int y1,int r,int value)
  {
        int x=0,y=r,p;
        p=1-r;
        circlepoints(x1,y1,x,y,value);
        while(x<y)
        {
          if(p<0)
          {
                p=p+2.0*x+3.0;
          }
          else
          {
                p=p+2.0*(x-y)+5.0;
```

```
                y=y-1;
            }
            x=x+1;
            circlepoints(x1,y1,x,y,value);
        }
}

void main()
{
        int gd =DETECT,a,gm;
        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        printf("enter radius");
        scanf("%d",&a);
        midpointcircle(100,100,a,5);
        getch();
        closegraph();
}
```

3. Write a program to clip a line using Cohen and Sutherland lineClipping algorithm.

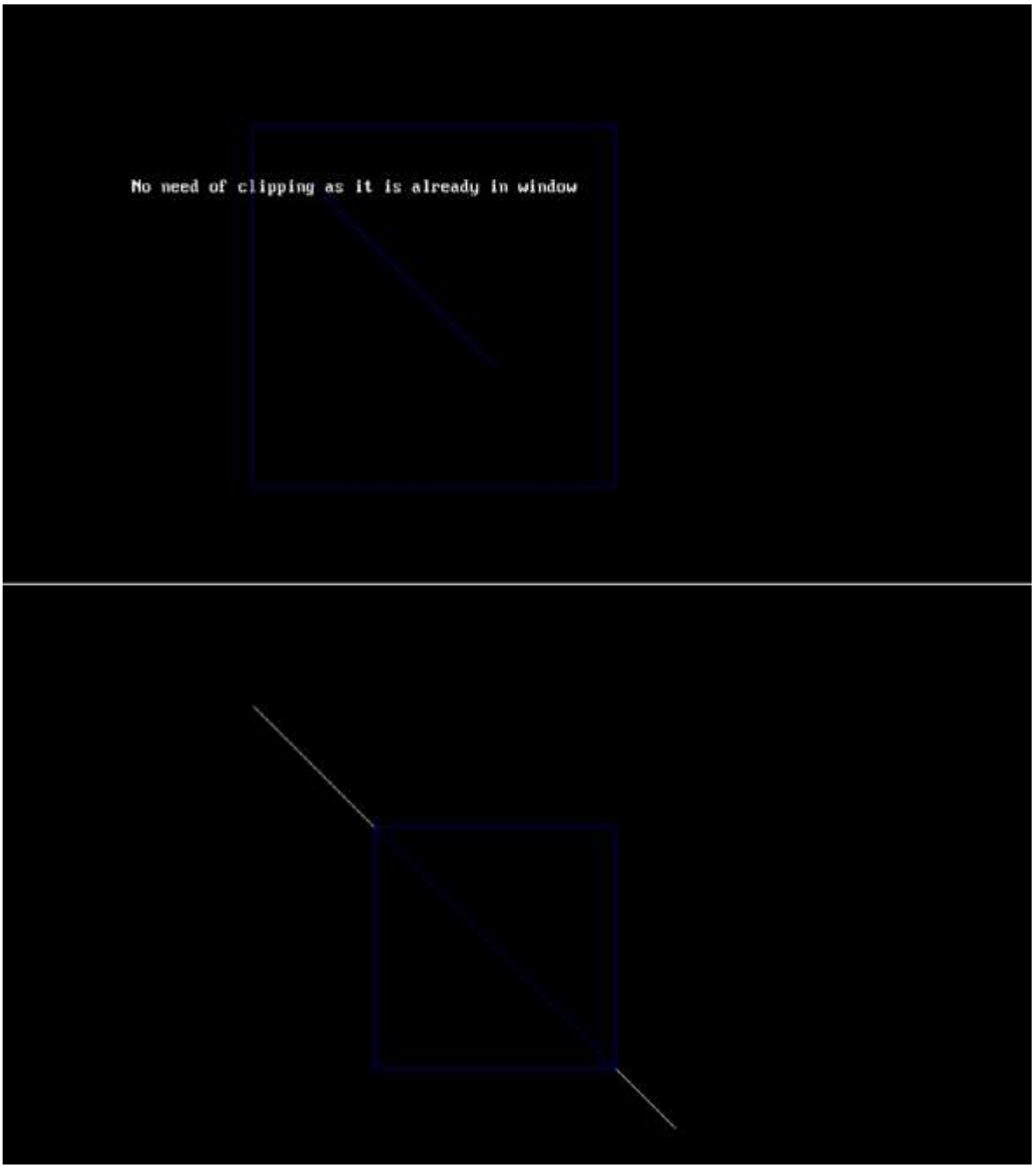
```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
int rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4];
int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;
float slope;
int x,y,x1,y1,i;
int gm=DETECT,gd;
initgraph(&gm,&gd,"c:\\turboc3\\bgi");
printf("\n****** Cohen Sutherlsnd Line Clipping algorithm **********");
printf("\n Now, enter XMin, YMin =");
scanf("%d %d",&W_xmin,&W_ymin);
printf("\n First enter XMax, YMax =");
scanf("%d %d",&W_xmax,&W_ymax);
printf("\n Please enter intial point x and y= ");
scanf("%d %d",&x,&y);
printf("\n Now, enter final point x1 and y1= ");
scanf("%d %d",&x1,&y1);cleardevice();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(x,y,x1,y1);
if(y>W_ymax) {
rcode_begin[0]=1;    // Top
flag=1 ;
}
if(y<W_ymin) {
rcode_begin[1]=1;         // Bottom
```

```
flag=1;
}
if(x>W_xmax) {
rcode_begin[2]=1;          // Right
flag=1;
}
if(x<W_xmin) {
rcode_begin[3]=1;          //Left
flag=1;
}

//end point of Line
if(y1>W_ymax){
rcode_end[0]=1;          // Top
flag=1;
}
if(y1<W_ymin) {
rcode_end[1]=1;          // Bottom
flag=1;
}
if(x1>W_xmax){
rcode_end[2]=1;          // Right
flag=1;
}
if(x1<W_xmin){
rcode_end[3]=1;          //Left
flag=1;
 }
if(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)  //left
{
y=y+(float) (W_xmin-x)*slope ;
x=W_xmin;

}
if(rcode_begin[2]==1 && rcode_begin[3]==0)       // right
{
y=y+(float) (W_xmax-x)*slope ;
x=W_xmax;

}
if(rcode_begin[0]==1 && rcode_begin[1]==0)      // top
{
x=x+(float) (W_ymax-y)/slope ;
y=W_ymax;

}
```

```
if(rcode_begin[0]==0 && rcode_begin[1]==1)     // bottom
{
x=x+(float) (W_ymin-y)/slope ;
y=W_ymin;

}
// end points
if(rcode_end[2]==0 && rcode_end[3]==1)  //left
{
y1=y1+(float) (W_xmin-x1)*slope ;
x1=W_xmin;

}
if(rcode_end[2]==1 && rcode_end[3]==0)       // right
{
y1=y1+(float) (W_xmax-x1)*slope ;
x1=W_xmax;

}
if(rcode_end[0]==1 && rcode_end[1]==0)      // top
{
x1=x1+(float) (W_ymax-y1)/slope ;
y1=W_ymax;

}
if(rcode_end[0]==0 && rcode_end[1]==1)     // bottom
{
x1=x1+(float) (W_ymin-y1)/slope ;
y1=W_ymin;

}
```

```
}
setcolor(BLUE);
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);line(x,y,x1,y1);
getch();
closegraph();
}
```



4. Write a program to clip a Polygon using Sutherland Hodgemanalgorithm.

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

int *x1,*y1,*x2,*y2,*x,*y,*ymax,*ymin,i,j,nin,nout,*pintersect;
float *dx,*xa;


int sign(long int a)
{
        if (a<0) return(-1);
        else if (a==0) return(0);
   else return(1);
}

void clip_polygon(void)
{
        int s1,s2,f1,f2,spcross,svisible;
        int visible(int a,int b,int c,int d,int e,int f);
        void output(int a,int b,int *c,int *d,int *e);
        int cross(int a,int b,int c,int d,int e,int f,int g,int h);
        int *intersect(int a,int b,int c,int d,int e,int f,int g,int h);
        pintersect=(int *)malloc(sizeof(int)*2);
        for (i=1;i<=4;i++)
        {
```

```
nout=0;
for (j=0;j<nin;j++)
        *(x2+j)=*(y2+j)=0;
for (j=1;j<=nin;j++)
{
        if (j!=1) {}
        else
        {
                f1=*(x1+j-1);
                f2=*(y1+j-1);
                s1=*(x1+j-1);
                s2=*(y1+j-1);
                svisible=visible(s1,s2,*(x+i-1),*(y+i-1),*(x+i),*(y+i));
                if (svisible>=0) output(s1,s2,&nout,x2,y2);
                        continue;
        }
        spcross=cross(s1,s2,*(x1+j-1),*(y1+j-1),*(x+i-1),*(y+i-1),*(x+i),*(y+i));
        if (!spcross) {}
        else
                {
                pintersect=intersect(s1,s2,*(x1+j-1),*(y1+j-1),*(x+i-1),*(y+i-
1),*(x+i),*(y+i));
                output(*pintersect,*(pintersect+1),&nout,x2,y2);
                }
        s1=*(x1+j-1);
        s2=*(y1+j-1);
        svisible=visible(s1,s2,*(x+i-1),*(y+i-1),*(x+i),*(y+i));
        if (svisible>=0) output(s1,s2,&nout,x2,y2);
}
if (!nout) continue;
spcross=cross(s1,s2,f1,f2,*(x+i-1),*(y+i-1),*(x+i),*(y+i));
```

```
        if (!spcross) {}
        else
        {
                pintersect=intersect(s1,s2,f1,f2,*(x+i-1),*(y+i-1),*(x+i),*(y+i));
                output(*pintersect,*(pintersect+1),&nout,x2,y2);
        }
        for (j=0;j<nout;j++)
        {
                *(x1+j)=*(x2+j);
                *(y1+j)=*(y2+j);
        }
        nin=nout;
        }
        *(x2+nout)=*x2;
        *(y2+nout)=*y2;
}

int cross(int s1,int s2,int p1,int p2,int wx1,int wy1,int wx2,int wy2)
{
        int pvisible1,pvisible2;
        int visible(int a,int b,int c,int d,int e,int f);
        pvisible1=visible(s1,s2,wx1,wy1,wx2,wy2);
        pvisible2=visible(p1,p2,wx1,wy1,wx2,wy2);
        if (pvisible1==-pvisible2)
                return 1;
   else return 0;
}

int visible(int sx1,int sx2,int px1,int py1,int px2,int py2)
{
        long int temp1,temp2,temp3;
```

```
        int pvisible;
        temp1=(long)(sx1-px1)*(long)(py2-py1);
        temp2=(long)(sx2-py1)*(long)(px2-px1);
        temp3=temp2-temp1;
        pvisible=sign(temp3);
        return (pvisible);
}

int *intersect(int px1,int py1,int px2,int py2,int wx1,int wy1,int wx2,int wy2)
{
        float parameter[2][1],coeff[2][2],temp1,temp2;
        int right[2][1];
        coeff[0][0]=px2-px1;
        coeff[0][1]=wx1-wx2;
        coeff[1][0]=py2-py1;
        coeff[1][1]=wy1-wy2;
        right[0][0]=wx1-px1;
        right[1][0]=wy1-py1;
        temp1=(coeff[0][0]*coeff[1][1])-(coeff[0][1]*coeff[1][0]);
        temp2=coeff[0][0];
        coeff[0][0]=(coeff[1][1])/temp1;
        coeff[1][1]=temp2/temp1;
        coeff[0][1]=-(coeff[0][1])/temp1;
        coeff[1][0]=-(coeff[1][0])/temp1;
        parameter[0][0]=(coeff[0][0]*right[0][0])+(coeff[0][1]*right[1][0]);
        parameter[1][0]=(coeff[1][0]*right[0][0])+(coeff[1][1]*right[1][0]);
        *pintersect=px1+(px2-px1)*parameter[0][0];
        *(pintersect+1)=py1+(py2-py1)*parameter[0][0];
        return(pintersect);
 }
```

```
void output(int vertex1,int vertex2,int *n,int *x2,int *y2)
{
        (*n)++;
        *(x2+(*n)-1)=vertex1;
        *(y2+(*n)-1)=vertex2;
}

void include(int *end_edge,int final_edge,int scan)
{
        while((*end_edge<=final_edge)&&(*(ymax+*end_edge)>=scan))
        (*end_edge)++;
}

void fillscan(int end_edge,int start_edge,int scan)
{
        int nx,j,k;
        nx=(end_edge-start_edge)/2;
        j=start_edge;
        for (k=1;k<=nx;k++)
        {
        line(*(xa+j),scan,*(xa+j+1),scan);
        j+=2;
        }
}

void update_xvalues(int last_edge,int *start_edge,int scan)
{
        int k1,k2;
        k2=last_edge;
        for (k1=last_edge;k1>=*start_edge;k1--)
        {
```

```
        if (*(ymin+k1)<scan)
        {
                *(xa+k2)=*(xa+k1)+*(dx+k1);
                if (k1!=k2)
                {
                                *(ymin+k2)=*(ymin+k1);
                                *(dx+k2)=*(dx+k1);
                }
                k2--;
        }
        }
        *start_edge=k2+1;
}

void xsort(int start_edge,int last_edge)
{
        int k,l;
        float t;
        for (k=start_edge;k<=last_edge;k++)
        {
        l=k;
        while((l>start_edge)&&(*(xa+l)<*(xa+l-1)))
        {
                t=*(ymin+l);
                *(ymin+l)=*(ymin+l-1);
                *(ymin+l-1)=t;
                t=*(xa+l);
                *(xa+l)=*(xa+l-1);
                *(xa+l-1)=t;
                t=*(dx+l);
                *(dx+l)=*(dx+l-1);
```

```
                *(dx+l-1)=t;
                l--;
        }
  }
}

void poly_insert(int j,int xc1,int yc1,int xc2,int yc2)
{
        int j1,ym;
        j1=j;
        if (yc1>yc2) ym=yc1;
        else ym=yc2;
        while((j1!=0)&&(*(ymax+j1-1)<ym))
        {
        *(ymax+j1)=*(ymax+j1-1);
        *(ymin+j1)=*(ymin+j1-1);
        *(xa+j1)=*(xa+j1-1);
        *(dx+j1)=*(dx+j1-1);
        j1--;
        }
        *(ymax+j1)=ym;
        *(dx+j1)=-(float)(xc2-xc1)/(yc2-yc1);
        if (yc1>yc2)
        {
           *(ymin+j1)=yc2;
           *(xa+j1)=xc1;
        }
        else
   {
        *(ymin+j1)=yc1;
        *(xa+j1)=xc2;
```

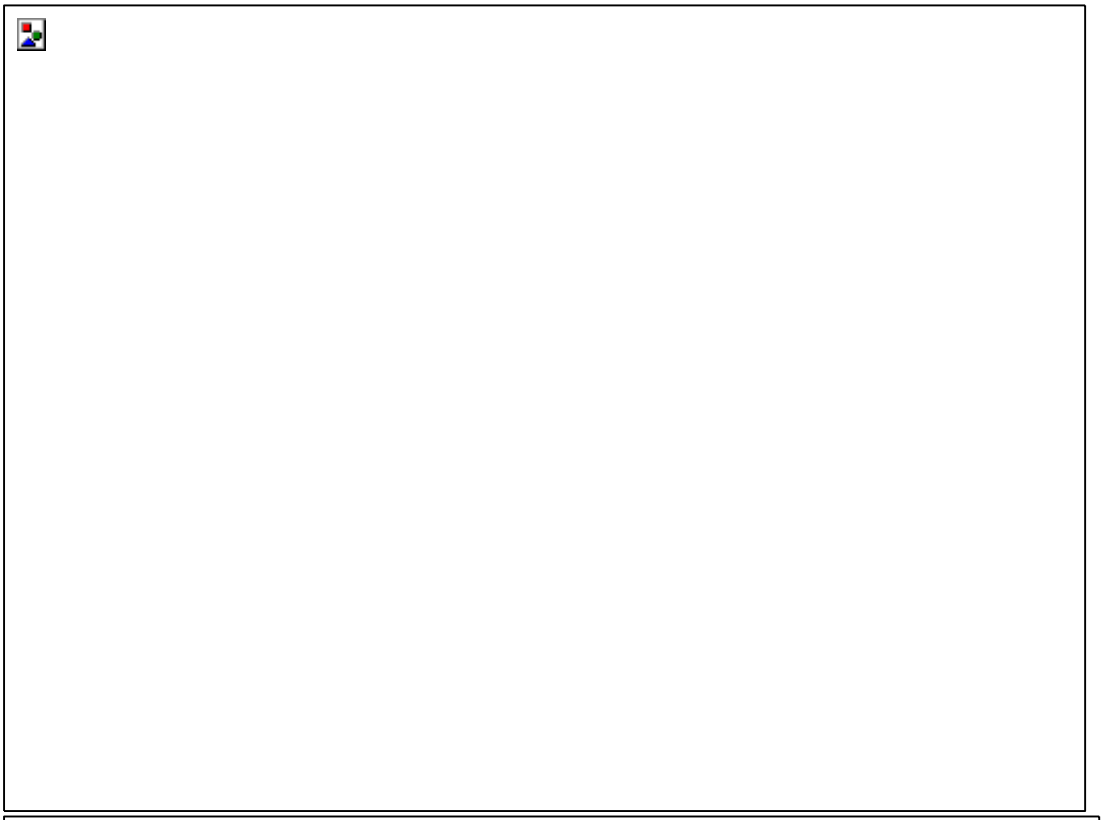
```
   }
}

void getpoint(int i,int *xtemp,int *ytemp)
{
        *xtemp=*(x2+i);
        *ytemp=*(y2+i);
}

void loadpolygon(int i,int *edges)
{
        int xc1,xc2,yc1,yc2,i1,k;
        getpoint(i,&xc1,&yc1);
        i1=i+1;
        *edges=0;
        for(k=1;k<=nin;k++)
        {
           getpoint(i1,&xc2,&yc2);
        if (yc1==yc2)
        xc1=xc2;
        else
        {
                poly_insert(*edges,xc1,yc1,xc2,yc2);
                        (*edges)++;
                yc1=yc2;
                xc1=xc2;
        }
        i1++;
        }
        (*edges)--;
}
```

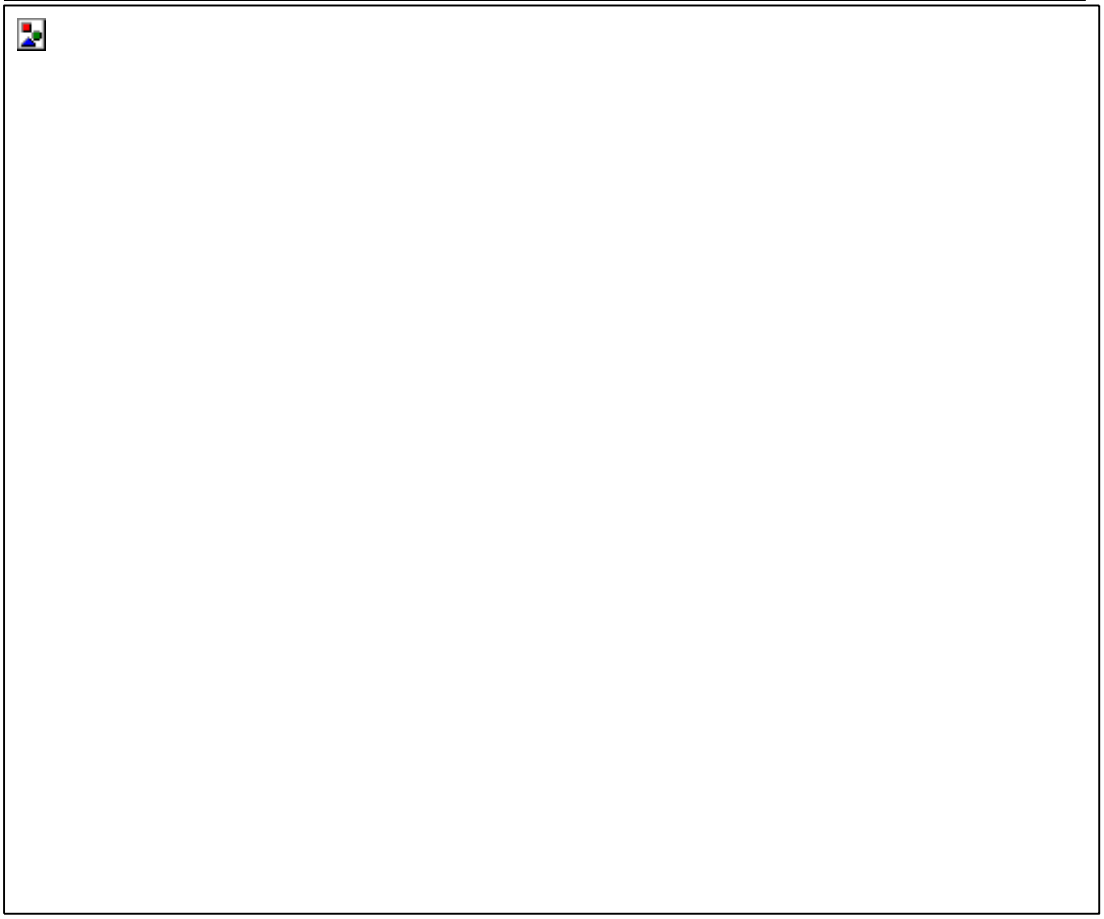
```
void fillpolygon(int index)
{
        int edges,scan,start_edge,end_edge;
        loadpolygon(index,&edges);
        if (edges<1) return;
        scan=*ymax;
        start_edge=0;
        end_edge=0;
        include(&end_edge,edges,scan);
        while(end_edge!=start_edge)
        {
        xsort(start_edge,end_edge-1);
        fillscan(end_edge,start_edge,scan);
        scan--;
        update_xvalues(end_edge-1,&start_edge,scan);
        include(&end_edge,edges,scan);
        }
}

void main()
{
        int gd=DETECT,gm;
        int gdriver = DETECT,gmode,errorcode;
        clrscr();
        /*  //Request autodetection
        int gdriver = DETECT,gmode,errorcode;
         //int xmax,ymax,x1,y1,x2,y2,l;
        //Initialize graphics and local variables
        initgraph(&gdriver,&gmode,"C:\\TURBOC3\\BGI");
        //Read result of initialization
```

```
errorcode=graphresult();
if(errorcode!=grOk) //Error occured
{
printf("Graphics error : %s\n",grapherrormsg(errorcode));
printf("Press any key to halt.");
getch();
exit(1);
}*/

x=(int *)malloc(sizeof(int)*5);
y=(int *)malloc(sizeof(int)*5);
printf("Enter number of sides in polygon : ");
scanf("%d",&nin);
x1=(int *)malloc(sizeof(int)*2*nin);
y1=(int *)malloc(sizeof(int)*2*nin);
x2=(int *)malloc(sizeof(int)*2*nin);
y2=(int *)malloc(sizeof(int)*2*nin);
ymax=(int *)malloc(sizeof(int)*2*nin);
ymin=(int *)malloc(sizeof(int)*2*nin);
xa=(float *)malloc(sizeof(float)*2*nin);
dx=(float *)malloc(sizeof(float)*2*nin);
printf("Enter the coordinates of the polygon vertices (x y) :\n");
for (i=0;i<nin;i++)
{
printf("%d",(i+1));
printf(":");
scanf("%d%d",&*(x1+i),&*(y1+i));
}
*(x1+nin)=*x1;
*(y1+nin)=*y1;
printf("\n\nEnter the coordinates of the window vertices :\n");
```

```
for (i=0;i<4;i++)
{
   printf("%d",(i+1));
   printf(":");
   scanf("%d%d",&*(x+i),&*(y+i));
}
*(x+4)=*x;
*(y+4)=*y;
 // registerbgidriver(EGAVGA_driver);
initgraph(&gdriver,&gmode,"C:\\TURBOC3\\BGI");
errorcode = graphresult();
// initgraph(&gd,&gm,"");
printf("Before clipping");
outtextxy(*x1+10,*y1-10,"Polygon");
outtextxy(*(x+1)+10,*(y+1)-10,"Clipping Window");
for (i=0;i<4;i++)
          line(*(x+i),*(y+i),*(x+i+1),*(y+i+1));
for (i=0;i<nin;i++)
          line(*(x1+i),*(y1+i),*(x1+i+1),*(y1+i+1));
getch();
clearviewport();
printf("After clipping");
//rectangle(200,200,400,400);
for (i=0;i<4;i++)
          line(*(x+i),*(y+i),*(x+i+1),*(y+i+1));
clip_polygon();
for (i=0;i<nin;i++)
{
   if(*(y2+i)<*(y2+i+1))
{
          *(ymax+i)=*(y2+i+1);
```

```
                *(ymin+i)=*(y2+i);
                *(xa+i)=*(x2+i+1);
        }
        else
        {
                *(ymax+i)=*(y2+i);
                *(ymin+i)=*(y2+i+1);
                *(xa+i)=*(x2+i);
        }
        *(dx+i)=*(y2+i)-*(y2+i+1);
        if (*(dx+i))
                *(dx+i)=(float)(*(x2+i)-*(x2+i+1))/(*(dx+i));
        line(*(x2+i),*(y2+i),*(x2+i+1),*(y2+i+1));
        }
        fillpolygon(0);
        getch();
        closegraph();
}
```



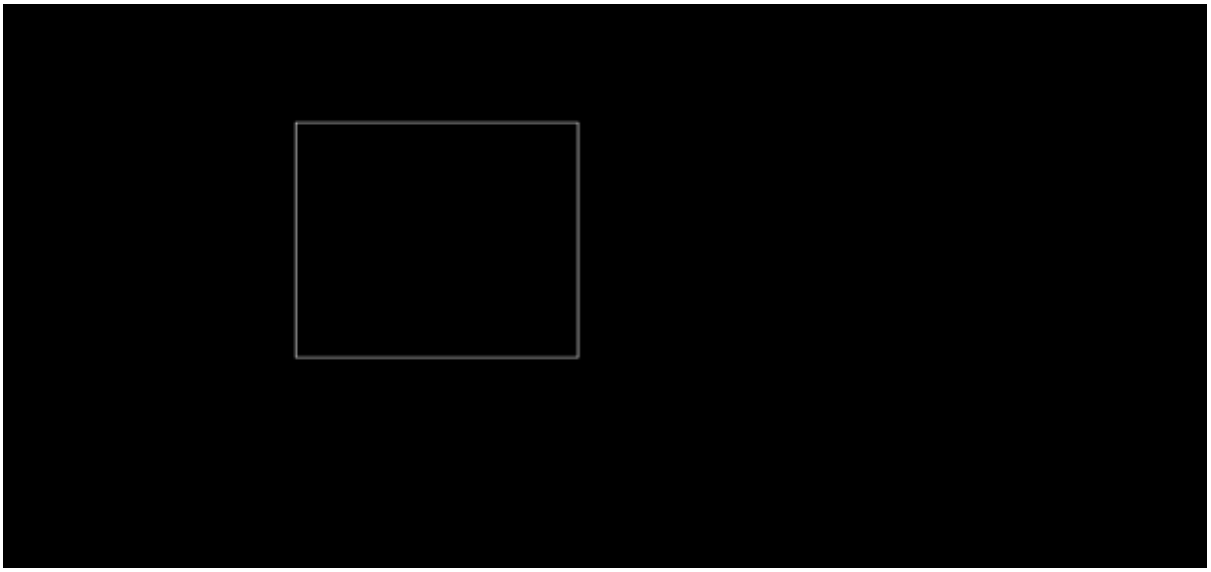


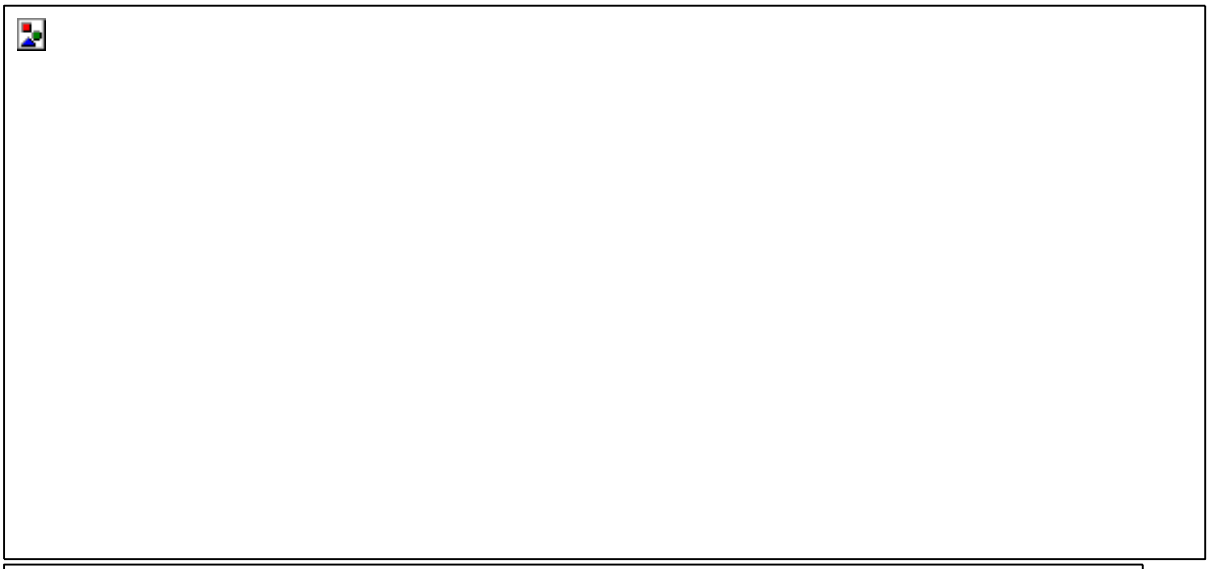
5. Write a program to fill a polygon using Scan line fill algorithm.

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

main()
{

int n,i,j,k,gd,gm,dy,dx;
int x,y,temp;
int a[20][2],xi[20];
float slope[20];



clrscr();
printf("\n\n\tEnter the no. of edges of polygon : ");
scanf("%d",&n);
printf("\n\n\tEnter the cordinates of polygon :\n\n\n ");

for(i=0;i<n;i++)
{
printf("\tX%d Y%d : ",i,i);
scanf("%d %d",&a[i][0],&a[i][1]);
}

a[n][0]=a[0][0];
a[n][1]=a[0][1];



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

```
/*- draw polygon -*/

for(i=0;i<n;i++)
{
line(a[i][0],a[i][1],a[i+1][0],a[i+1][1]);
}

getch();



for(i=0;i<n;i++)
{
dy=a[i+1][1]-a[i][1];
dx=a[i+1][0]-a[i][0];

if(dy==0) slope[i]=1.0;
if(dx==0) slope[i]=0.0;

if((dy!=0)&&(dx!=0)) /*- calculate inverse slope -*/
{
slope[i]=(float) dx/dy;
}
}

for(y=0;y< 480;y++)
{
k=0;
for(i=0;i<n;i++)
{
```

```
if( ((a[i][1]<=y)&&(a[i+1][1]>y))||
((a[i][1]>y)&&(a[i+1][1]<=y)))
{
xi[k]=(int)(a[i][0]+slope[i]*(y-a[i][1]));
k++;
}
}

for(j=0;j<k-1;j++) /*- Arrange x-intersections in order -*/
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(35);
for(i=0;i<k;i+=2)
{
line(xi[i],y,xi[i+1]+1,y);
getch();
}
}
}
```





6. Write a program to apply various 2D transformations on a 2Dobject(use homogeneous coordinates).

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#define pi (22/7)
#define sz 3
double x[3][3],res[3][3],tm[3][3];
int t2[3][2];

void prod(double a[sz][sz],double b[sz][sz],double c[sz][sz],int r1,int c1,int c2)
{	int i,j,k;
	for(i=0;i<r1;i++)
	{
		for(j=0;j<c2;j++)
		{
			for(k=0;k<c1;k++)
			{
				c[i][j] += a[i][k]*b[k][j];
			}
		}
	}
}
void drawtriangle(int t[3][2])
{	int i;
	for(i=0;i<3;i++)
	{
		line(t[i][0]+100,100-t[i][1],t[(i+1)%3][0]+100,100-t[(i+1)%3][1]);
	}
	line(0,100,400,100);
	line(100,0,100,400);
```

```
}
void translation(int t1[3][2])
{
	int i,j,val;
	for(i=0;i<3;i++)
	{
		for(j=0;j<2;j++)
		{
			x[i][j] = t1[i][j];
			t2[i][j] = 0;
			res[i][j] = 0;
			if(i == j)
				tm[i][j] = 1;
			else
				tm[i][j] = 0;
		}
		x[i][j] = 1;
		res[i][j] = 0;
		if(i == j)
			tm[i][j] = 1;
		else
			tm[i][j] = 0;
	}
	printf("\n Shift in x-axis : ");
	scanf("%d",&val);
	tm[2][0] = val;
	printf("\n Shift in y-axis : ");
	scanf("%d",&val);
	tm[2][1] = val;
	prod(x,tm,res,3,3,3);
	for(i=0;i<3;i++)
```

```
        {
                for(j=0;j<2;j++)
                {
                        t2[i][j] = (int)res[i][j];
                }
        }
        settextstyle(2,0,5);
        outtextxy(10,405,"Press any key to continue...");
        getch();
        clrscr();
        outtextxy(10,10,"\n Triangle after transformation ");
        drawtriangle(t2);
        outtextxy(10,405,"Press any key to continue...");
        getch();
}
void rotation(int t1[3][2])
{
        int i,j,ang;
        double sinx[] = {0,0.5,0.7,0.8,1,0,-1,0};
        double cosx[] = {1,0.8,0.7,0.5,0,-1,0,1};
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        x[i][j] = t1[i][j];
                        t2[i][j] = 0;
                        res[i][j] = 0;
                        if(i == j)
                                tm[i][j] = 1;
                        else
                                tm[i][j] = 0;
```

```
                }
                x[i][j] = 1;
                res[i][j] = 0;
                if(i == j)
                        tm[i][j] = 1;
                else
                        tm[i][j] = 0;
        }
        printf("\n Choose the angle of rotation : ");
        printf("\n\t 1. 0 degree\n\t 2. 30 degree\n\t 3. 45 degree\n\t 4. 60 degree\n\t 5. 90
degree");
        printf("\n\t 6. 180 degree\n\t 7. 270 degree\n\t 8. 360 degree\n Enter your choice : ");
        scanf("%d",&ang);
        if(ang > 0 && ang < 9)
        {
                tm[0][0] = tm[1][1] = cosx[ang-1];
                tm[0][1] = tm[1][0] = sinx[ang-1];
        }
        else
        {
                tm[0][0] = tm[1][1] = cosx[0];
                tm[0][1] = tm[1][0] = sinx[0];
        }
        tm[1][0] *= (-1);
        prod(x,tm,res,3,3,3);
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        t2[i][j] = (int)res[i][j];
                }
```

```
        }
        settextstyle(2,0,5);
        outtextxy(10,405,"Press any key to continue...");
        getch();
        clrscr();
        outtextxy(10,10,"\n Triangle after transformation ");
        drawtriangle(t2);
        outtextxy(10,405,"Press any key to continue...");
        getch();
}
void scaling(int t1[3][2])
{
        int i,j;
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        x[i][j] = t1[i][j];
                        t2[i][j] = 0;
                        res[i][j] = 0;
                        if(i == j)
                                tm[i][j] = 1;
                        else
                                tm[i][j] = 0;
                }
                x[i][j] = 1;
                res[i][j] = 0;
                if(i == j)
                        tm[i][j] = 1;
                else
                        tm[i][j] = 0;
```

```
        }
        printf("\n Enter the Scaling value : ");
        scanf("%lf",&tm[0][0]);
        tm[1][1] = tm[0][0];
        prod(x,tm,res,3,3,3);
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        t2[i][j] = (int)res[i][j];
                }
        }
        settextstyle(2,0,5);
        outtextxy(10,405,"Press any key to continue...");
        getch();
        clrscr();
        outtextxy(10,10,"\n Triangle after transformation ");
        drawtriangle(t2);
        outtextxy(10,405,"Press any key to continue...");
        getch();
}
void reflection(int t1[3][2])
{
        int i,j,ch;
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        x[i][j] = t1[i][j];
                        t2[i][j] = 0;
                        res[i][j] = 0;
```

```
                        if(i == j)
                                tm[i][j] = 1;
                        else
                                tm[i][j] = 0;
                }
                x[i][j] = 1;
                res[i][j] = 0;
                if(i == j)
                        tm[i][j] = 1;
                else
                        tm[i][j] = 0;
        }
        printf("\n Type of Reflection : ");
        printf("\n\t 1. Along x-axis ; y = 0\n\t 2. Along y-axis ; x = 0\n\t 3. Along y = x\n\t 4. Along y =
-x ");
        printf("\n Enter your choice : ");
        scanf("%d",&ch);
        if(ch > 0 && ch < 4)
        {               if(ch < 3)
                {
                        tm[0][1] = tm[1][0] = 0;
                        if(ch == 1)
                                tm[1][1] = -1;
                        else
                                tm[0][0] = -1;
                }
                else
                {
                        tm[0][0] = tm[1][1] = 0;
                        if(ch == 3)
                                tm[0][1] = tm[1][0] = 1;
```

```
                        else
                                tm[0][1] = tm[1][0] = -1;
                }
        }
        else // taking the case of reflection along y = 0
        {
                tm[0][1] = tm[1][0] = 0;
                tm[1][1] = -1;
        }
        prod(x,tm,res,3,3,3);
        for(i=0;i<3;i++)
        {
                for(j=0;j<2;j++)
                {
                        t2[i][j] = (int)res[i][j];
                }
        }
        settextstyle(2,0,5);
        outtextxy(10,405,"Press any key to continue...");
        getch();
        clrscr();
        outtextxy(10,10,"\n Triangle after transformation ");
        drawtriangle(t2);
        outtextxy(10,405,"Press any key to continue...");
        getch();
}
void distortion(int t1[3][2])
{
        int i,j;
        for(i=0;i<3;i++)
        {
```

```
		for(j=0;j<2;j++)
		{
			x[i][j] = t1[i][j];
			t2[i][j] = 0;
			res[i][j] = 0;
			if(i == j)
				tm[i][j] = 1;
			else
				tm[i][j] = 0;
		}
		x[i][j] = 1;
		res[i][j] = 0;
		if(i == j)
			tm[i][j] = 1;
		else
			tm[i][j] = 0;
}
printf("\n Enter the Scaling value for x-axis : ");
scanf("%lf",&tm[0][0]);
printf("\n Enter the Scaling value for y-axis : ");
scanf("%lf",&tm[1][1]);
prod(x,tm,res,3,3,3);
for(i=0;i<3;i++)
{
		for(j=0;j<2;j++)
		{
			t2[i][j] = (int)res[i][j];
		}
}
settextstyle(2,0,5);
outtextxy(10,405,"Press any key to continue...");
```

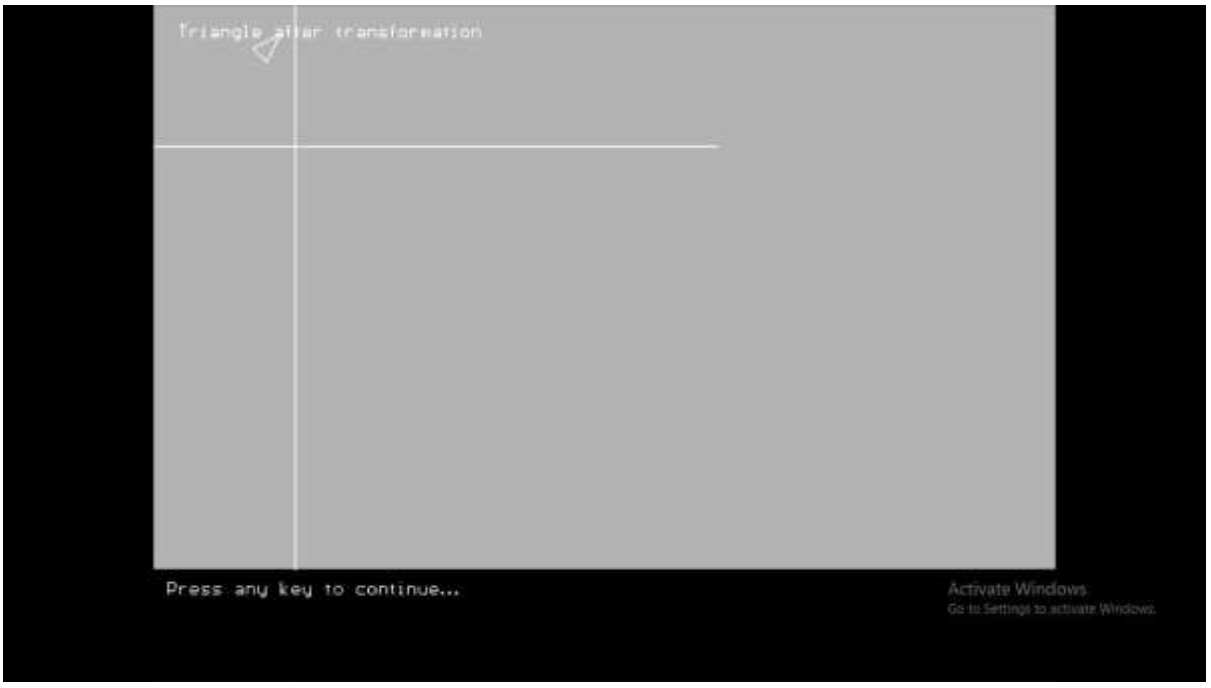
```
        getch();
        clrscr();
        outtextxy(10,10,"\n Triangle after transformation ");
        drawtriangle(t2);
        outtextxy(10,405,"Press any key to continue...");
        getch();
}
void main()
{
        int gd = DETECT,gm,i,ch;
        int t1[3][2];
        initgraph(&gd,&gm,"C:\\TURBOC3\\bgi");
        clrscr();
        for(i=0;i<3;i++)
        {
                printf("\n Enter the value of x%d : ",i+1);
                scanf("%d",&t1[i][0]);
                printf("\n Enter the value of y%d : ",i+1);
                scanf("%d",&t1[i][1]);
        }
        settextstyle(2,0,5);
        outtextxy(10,405,"Press any key to continue...");
        getch();
        clrscr();
        outtextxy(10,10,"Triangle before transformation ");
        drawtriangle(t1);
        outtextxy(10,405,"Press any key to continue...");
        getch();
        clrscr();
        do
        {
```

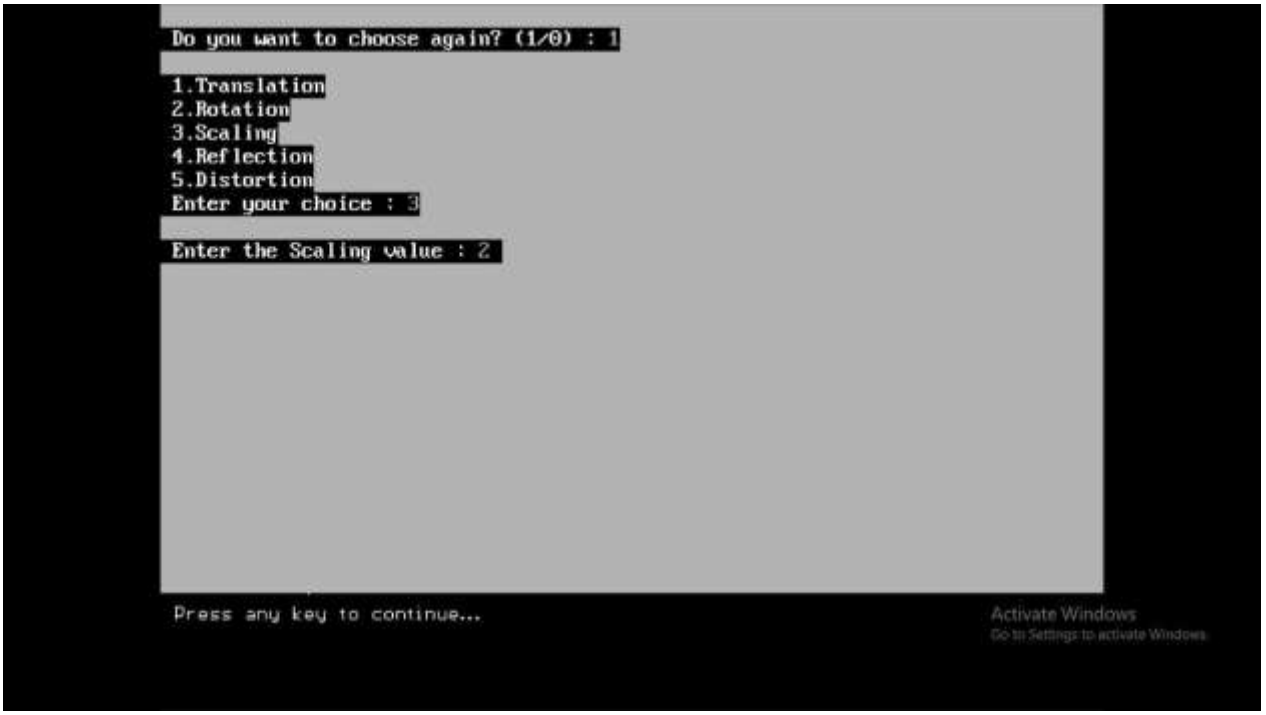
```
		printf("\n 1.Translation\n 2.Rotation\n 3.Scaling\n 4.Reflection\n 5.Distortion\n
Enter your choice : ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:
				translation(t1);
				break;
			case 2:
				rotation(t1);
				break;
			case 3:
				scaling(t1);
				break;
			case 4:
				reflection(t1);
				break;
			case 5:
				distortion(t1);
				break;
			default:
				printf("\n Invalid Choice !");
		}
		clrscr();
		printf("\n Do you want to choose again? (1/0) : ");
		scanf("%d",&ch);
	}while(ch == 1);
	getch();
	closegraph();
	restorecrtmode();
}
```


Triangle before transformation
Press any key to continue...
Activate Windows
Go to Settings to activate Windows.

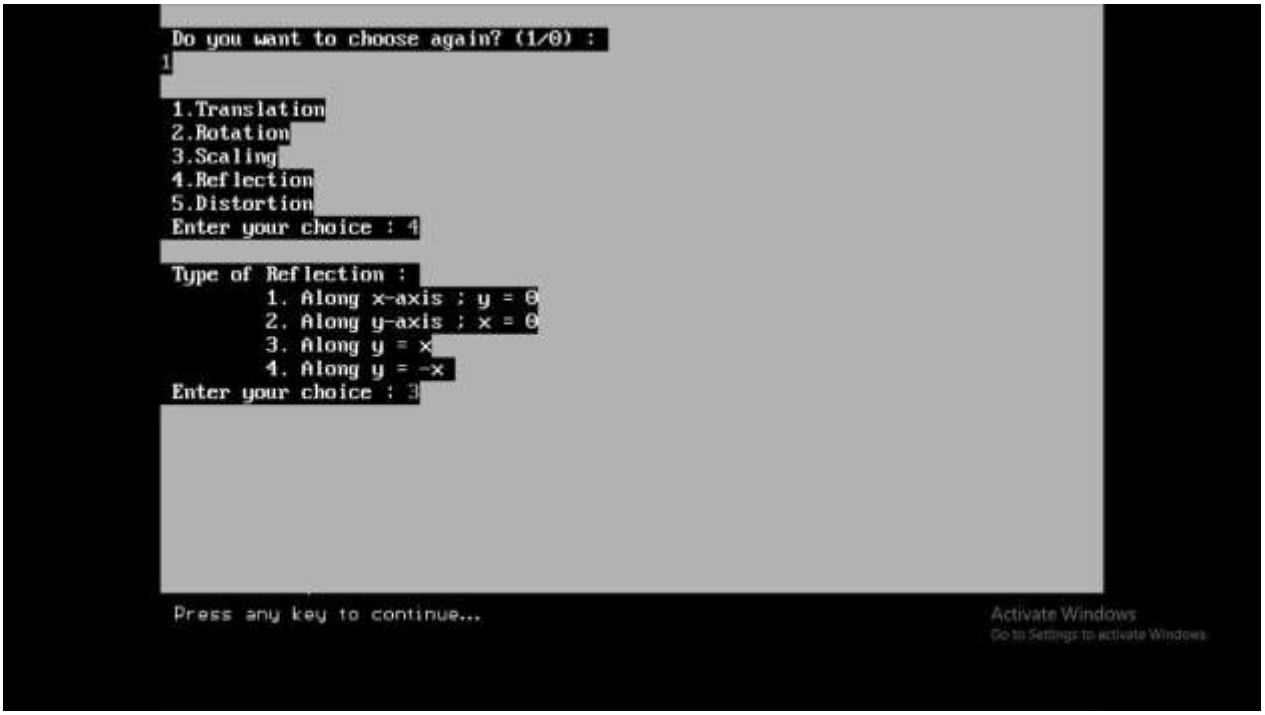

1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Distortion
Enter your choice : 1
Shift in x-axis : 2
Shift in y-axis : -2
Press any key to continue...
Activate Windows
Go to Settings to activate Windows.

When we rotate it 90 degree




Do you want to choose again? (1/0) : 1
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Distortion
Enter your choice : 3
Enter the Scaling value : 2
Press any key to continue...
Activate Windows
Go to Settings to activate Windows.


Triangle after transformation
Press any key to continue...
Activate Windows
Go to Settings to activate Windows.


Do you want to choose again? (1/0) :
1
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Distortion
Enter your choice : 4
Type of Reflection :
1. Along x-axis ; y = 0
2. Along y-axis ; x = 0
3. Along y = x
4. Along y = -x
Enter your choice : 3
Press any key to continue...
Activate Windows
Go to Settings to activate Windows.


Triangle after transformation
Press any key to continue...
Activate Windows
Go to Settings to activate Windows.


Do you want to choose again? (1/0) : 1
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Distortion
Enter your choice : 5
Enter the Scaling value for x-axis : 2
Enter the Scaling value for y-axis : 3
Press any key to continue...
Activate Windows
Go to Settings to activate Windows.
Triangle after transformation
Press any key to continue...
Activate Windows
Go to Settings to activate Windows.

7. Write a program to apply various 3D transformations on a 3D object and then apply parallel and perspective projection on it.

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

#define pi (22/7)
#define row 8
#define col 4

double x[8][4],res[8][4],tm[4][4];
int t2[8][4];

void prod(double a[row][col],double b[col][col],double c[row][col],int r1,int c1,int c2)
{
	int i,j,k;
	for(i=0;i<r1;i++)
	{
		for(j=0;j<c2;j++)
		{
			for(k=0;k<c1;k++)
			{
				c[i][j] += a[i][k]*b[k][j];
			}
		}
	}
}
void printmat(int arr[8][4])
{
	int i,j;
	for(i=0;i<8;i++)
	{
```

```
                printf("\n ");
                for(j=0;j<4;j++)
                {
                        printf("%d ",arr[i][j]);
                }
        }
        printf("\n Press any key to continue...");
        getch();
        clrscr();
        cleardevice();
}
void drawcuboid(int t1[8][4])
{
        int i,j;
        int t[8][4];
        for(i=0;i<8;i++)
        {
                for(j=0;j<4;j++)
                {
                        t[i][j] = t1[i][j];
                }
        }
        for(i=0;i<4;i++)
        {
                for(j=0;j<3;j++)
                {
                        t[i][j] += 20;
                }
        }
        j = 4;
        for(i=0;i<4;i++)
```

```
        {
                line(t[i][0],t[i][1],t[(i+1)%4][0],t[(i+1)%4][1]); // Forming sq ABCD
                line(t[j][0],t[j][1],t[((j+1)%4)+4][0],t[((j+1)%4)+4][1]); // Forming sq EFGH
                line(t[i][0],t[i][1],t[j][0],t[j][1]);                  // Joining AE,BF,CG,DH
                j++;
        }
        settextstyle(2,0,5);
        outtextxy(10,405,"Press any key to continue...");
        getch();
        clrscr();
        cleardevice();
}
void translation(int t1[8][4])
{
        int i,j,val;
        for(i=0;i<8;i++)
        {
                for(j=0;j<3;j++)
                {
                        x[i][j] = t1[i][j];
                        t2[i][j] = 0;
                        res[i][j] = 0;
                        if(i == j)
                                tm[i][j] = 1;
                        else
                                tm[i][j] = 0;
                }
                x[i][j] = 1;
                res[i][j] = 0;
                if(i == j)
                        tm[i][j] = 1;
```

```
                else
                        tm[i][j] = 0;
        }
        printf("\n Shift in x-axis : ");
        scanf("%d",&val);
        tm[3][0] = val;
        printf("\n Shift in y-axis : ");
        scanf("%d",&val);
        tm[3][1] = val;
        printf("\n Shift in z-axis : ");
        scanf("%d",&val);
        tm[3][2] = val;
        prod(x,tm,res,row,col,col);
        for(i=0;i<8;i++)
        {
                for(j=0;j<3;j++)
                {
                        t2[i][j] = (int)res[i][j];
                }
        }
        printf("\n Matrix X\' : ");
        printmat(t2);
        printf("\n Cuboid after transformation ");
        drawcuboid(t2);
}
void rotation(int t1[8][4])
{
        int i,j,ang,ch;
        double sinx[] = {0,0.5,0.7,0.8,1,0,-1,0};
        double cosx[] = {1,0.8,0.7,0.5,0,-1,0,1};
        for(i=0;i<8;i++)
```

```
	{
		for(j=0;j<3;j++)
		{
			x[i][j] = t1[i][j];
			t2[i][j] = 0;
			res[i][j] = 0;
			if(i == j)
				tm[i][j] = 1;
			else
				tm[i][j] = 0;
		}
		x[i][j] = 1;
		res[i][j] = 0;
		if(i == j)
			tm[i][j] = 1;
		else
			tm[i][j] = 0;
	}
	printf("\n Choose the type of rotation : \n\t 1. Rotation about x-axis\n\t 2. Rotation about y-axis");
	printf("\n\t 3. Rotation about z-axis\n Enter your choice : ");
	scanf("%d",&ch);
	printf("\n Choose the angle of rotation : ");
	printf("\n\t 1. 0 degree\n\t 2. 30 degree\n\t 3. 45 degree\n\t 4. 60 degree\n\t 5. 90 degree");
	printf("\n\t 6. 180 degree\n\t 7. 270 degree\n\t 8. 360 degree\n Enter your choice : ");
	scanf("%d",&ang);
	if(ch <= 1) // x-axis
	{
		if(ang > 0 && ang < 9)
		{
			tm[2][2] = tm[1][1] = cosx[ang-1];
```

```
                tm[1][2] = tm[2][1] = sinx[ang-1];
        }
        else
        {
                tm[2][2] = tm[1][1] = cosx[0];
                tm[1][2] = tm[2][1] = sinx[0];
        }
        tm[2][1] *= (-1);
}
else if(ch == 2) // y-axis
{
        if(ang > 0 && ang < 9)
        {
                tm[2][2] = tm[0][0] = cosx[ang-1];
                tm[0][2] = tm[2][0] = sinx[ang-1];
        }
        else
        {
                tm[2][2] = tm[1][1] = cosx[0];
                tm[1][2] = tm[2][1] = sinx[0];
        }
        tm[2][0] *= (-1);
}
else // z-axis
{
        if(ang > 0 && ang < 9)
        {
                tm[0][0] = tm[1][1] = cosx[ang-1];
                tm[0][1] = tm[1][0] = sinx[ang-1];
        }
        else
```

```
                {
                        tm[2][2] = tm[1][1] = cosx[0];
                        tm[1][2] = tm[2][1] = sinx[0];
                }
                tm[1][0] *= (-1);
        }
        prod(x,tm,res,row,col,col);
        for(i=0;i<8;i++)
        {
                for(j=0;j<3;j++)
                {
                        t2[i][j] = (int)res[i][j];
                }
        }
        printf("\n Matrix X\' : ");
        printmat(t2);
        printf("\n Cuboid after transformation ");
        drawcuboid(t2);
}
void scaling(int t1[8][4])
{
        int i,j;
        for(i=0;i<8;i++)
        {
                for(j=0;j<3;j++)
                {
                        x[i][j] = t1[i][j];
                        t2[i][j] = 0;
                        res[i][j] = 0;
                        if(i == j)
                                tm[i][j] = 1;
```

```
                else
                        tm[i][j] = 0;
        }
        x[i][j] = 1;
        res[i][j] = 0;
        if(i == j)
                tm[i][j] = 1;
        else
                tm[i][j] = 0;
    }
    printf("\n Enter the Scaling value : ");
    scanf("%lf",&tm[0][0]);
    tm[1][1] = tm[2][2] = tm[0][0];
    prod(x,tm,res,row,col,col);
    for(i=0;i<8;i++)
    {
        for(j=0;j<3;j++)
        {
            t2[i][j] = (int)res[i][j];
        }
    }
    printf("\n Matrix X\' : ");
    printmat(t2);
    printf("\n Cuboid after transformation ");
    drawcuboid(t2);
}
void reflection(int t1[8][4])
{
    int i,j,ch;
    for(i=0;i<8;i++)
    {
```

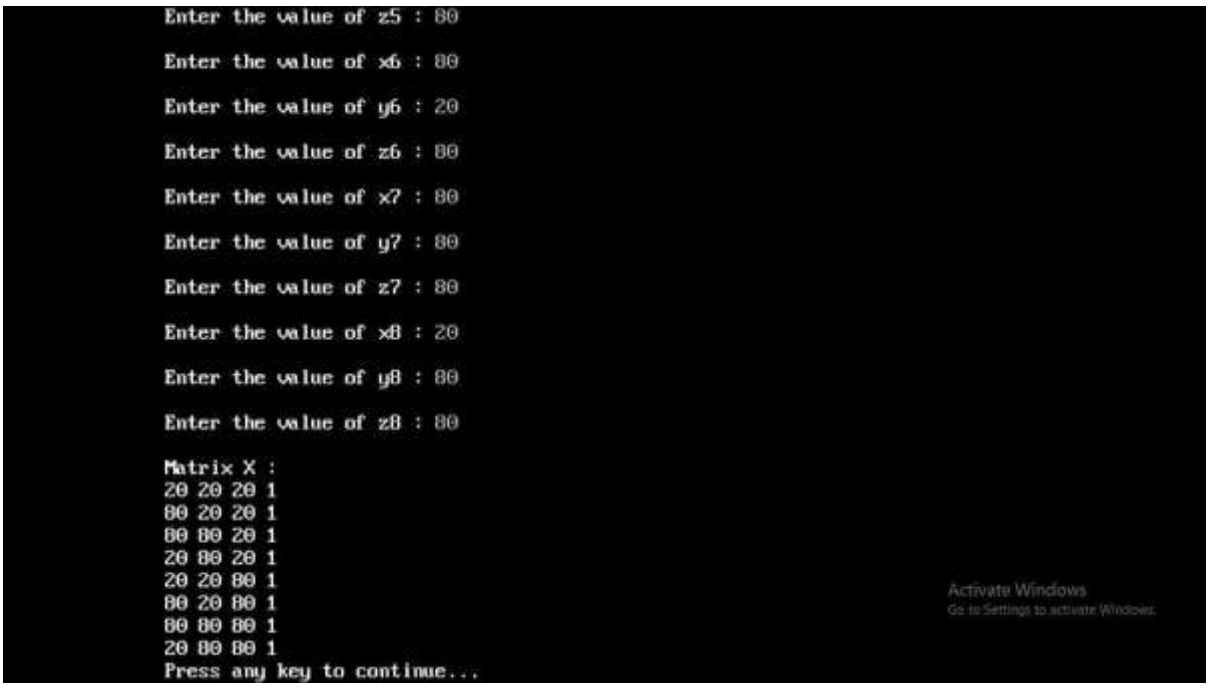
```
		for(j=0;j<3;j++)
		{
			x[i][j] = t1[i][j];
			t2[i][j] = 0;
			res[i][j] = 0;
			if(i == j)
				tm[i][j] = 1;
			else
				tm[i][j] = 0;
		}
		x[i][j] = 1;
		res[i][j] = 0;
		if(i == j)
			tm[i][j] = 1;
		else
			tm[i][j] = 0;
}
printf("\n Type of Reflection : ");
printf("\n\t 1. About xy plane\n\t 2. About yz plane\n\t 3. About xz plane");
printf("\n Enter your choice : ");
scanf("%d",&ch);
switch(ch)
{
		case 1:
			tm[2][2] = -1;
			break;
		case 2:
			tm[0][0] = -1;
			break;
		case 3:
			tm[1][1] = -1;
```

```
                        break;
                default : tm[2][2] = -1;
        }
        prod(x,tm,res,row,col,col);
        for(i=0;i<8;i++)
        {
                for(j=0;j<3;j++)
                {
                        t2[i][j] = (int)res[i][j];
                }
        }
        printf("\n Matrix X\' : ");
        printmat(t2);
        printf("\n Cuboid after transformation ");
        drawcuboid(t2);
}
void shearing(int t1[8][4])
{
        int i,j;
        for(i=0;i<8;i++)
        {
                for(j=0;j<3;j++)
                {
                        x[i][j] = t1[i][j];
                        t2[i][j] = 0;
                        res[i][j] = 0;
                        if(i == j)
                                tm[i][j] = 1;
                        else
                                tm[i][j] = 0;
                }
```

```
                x[i][j] = 1;
                res[i][j] = 0;
                if(i == j)
                        tm[i][j] = 1;
                else
                        tm[i][j] = 0;
        }
        printf("\n Enter the Scaling value x-axis, a : ");
        scanf("%lf",&tm[0][0]);
        printf("\n Enter the Scaling value y-axis, e (a != e) : ");
        scanf("%lf",&tm[1][1]);
        prod(x,tm,res,row,col,col);
        for(i=0;i<8;i++)
        {
                for(j=0;j<3;j++)
                {
                        t2[i][j] = (int)res[i][j];
                }
        }
        printf("\n Matrix X\' : ");
        printmat(t2);
        printf("\n Cuboid after transformation ");
        drawcuboid(t2);
}
void main()
{
        int i,j,ch,gd = DETECT,gm,t1[8][4];
        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        clrscr();
        cleardevice();
        for(i=0;i<8;i++)
```

```
	{
		printf("\n Enter the value of x%d : ",i+1);
		scanf("%d",&t1[i][0]);
		printf("\n Enter the value of y%d : ",i+1);
		scanf("%d",&t1[i][1]);
		printf("\n Enter the value of z%d : ",i+1);
		scanf("%d",&t1[i][2]);
		t1[i][3] = 1;
	}
	printf("\n Matrix X : ");
	printmat(t1);
	printf("\n Cuboid before transformation ");
	drawcuboid(t1);
	settextstyle(2,0,5);
	outtextxy(10,405,"Press any key to continue...");
	getch();
	do
	{
		clrscr();
		cleardevice();
		 printf("\n 1.Translation\n 2.Rotation\n 3.Scaling\n 4.Reflection\n 5.Shearing\n Enter
your choice : ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:
				translation(t1);
				break;
			case 2:
				rotation(t1);
				break;
```

```
                case 3:
                        scaling(t1);
                        break;
                case 4:
                        reflection(t1);
                        break;
                case 5:
                        shearing(t1);
                        break;
                default:
                        printf("\n Invalid Choice !");
        }
        printf("\n Do you want to choose again? (1/0) : ");
        scanf("%d",&ch);
}while(ch == 1);
getch();
closegraph();
restorecrtmode();
}
```



Cuboid before transformation

Press any key to continue...

1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Enter your choice : 1

Shift in x-axis : 2

Shift in y-axis : 3

Shift in z-axis : 2

Matrix X' :
22 23 22 0
82 23 22 0
82 83 22 0
22 83 22 0
22 23 82 0
82 23 82 0
82 83 82 0
22 83 82 0
Press any key to continue...

```
5.Shearing
Enter your choice : 2

Choose the type of rotation :
        1. Rotation about x-axis
        2. Rotation about y-axis
        3. Rotation about z-axis
Enter your choice : 1

Choose the angle of rotation :
        1. 0 degree
        2. 30 degree
        3. 45 degree
        4. 60 degree
        5. 90 degree
        6. 180 degree
        7. 270 degree
        8. 360 degree
Enter your choice : 3

Matrix X' :
20 0 28 0
80 0 28 0
80 42 70 0
20 42 70 0
20 -42 70 0
80 -42 70 0
80 0 112 0
20 0 112 0
Press any key to continue...
```



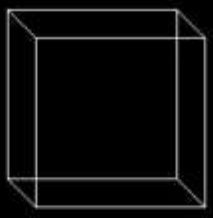
```
Cuboid after transformation
```

```
Press any key to continue...
```

```
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Enter your choice : 3

Enter the Scaling value : 2

Matrix X' :
40 40 40 0
160 40 40 0
160 160 40 0
40 160 40 0
40 40 160 0
160 40 160 0
160 160 160 0
40 160 160 0
Press any key to continue...
```



```
Cuboid after transformation
```



```
Press any key to continue...
```


1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Enter your choice : 4
Type of Reflection :
1. About xy plane
2. About yz plane
3. About xz plane
Enter your choice : 2
Matrix X' :
-20 20 20 0
-80 20 20 0
-80 80 20 0
-20 80 20 0
-20 20 80 0
-80 20 80 0
-80 80 80 0
-20 80 80 0
Press any key to continue...
Activate Windows
Go to Settings to activate Windows
Cuboid after transformation
Press any key to continue...
Activate Windows
Go to Settings to activate Windows

```
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Enter your choice : 5

Enter the Scaling value x-axis, a : 4

Enter the Scaling value y-axis, e (a != e) : 2

Matrix X' :
80 40 20 0
320 40 20 0
320 160 20 0
80 160 20 0
80 40 80 0
320 40 80 0
320 160 80 0
80 160 80 0
Press any key to continue...
```


Cuboid after transformation

Press any key to continue...

8. Write a program to draw Hermite/Bezier curve.

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

int x,y,z;

void main()
{
float u;
int gd,gm,ymax,i,n,c[4][3];



for(i=0;i<4;i++) { c[i][0]=0; c[i][1]=0; }




printf("\n\n Enter four points : \n\n");

for(i=0; i<4; i++)
{
printf("\t X%d Y%d : ",i,i);
scanf("%d %d",&c[i][0],&c[i][1]);

}

c[4][0]=c[0][0];
c[4][1]=c[0][1];

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

```
ymax = 480;

setcolor(13);
for(i=0;i<3;i++)
{
line(c[i][0],ymax-c[i][1],c[i+1][0],ymax-c[i+1][1]);
}

setcolor(3);
n=3;

for(i=0;i<=40;i++)
{
u=(float)i/40.0;
bezier(u,n,c);

if(i==0)
{ moveto(x,ymax-y);}
else
{ lineto(x,ymax-y); }
getch();
}
getch();
}
bezier(u,n,p)
float u;int n; int p[4][3];
{
int j;
float v,b;
float blend(int,int,float);
```

```
x=0;y=0;z=0;
for(j=0;j<=n;j++)
{
b=blend(j,n,u);
x=x+(p[j][0]*b);
y=y+(p[j][1]*b);
z=z+(p[j][2]*b);
}
}

float blend(int j,int n,float u)
{
int k;
float v,blend;
v=C(n,j);
for(k=0;k<j;k++)
{ v*=u; }
for(k=1;k<=(n-j);k++)
{ v *= (1-u); }
blend=v;
return(blend);
}

C(int n,int j)
{
int k,a,c;
a=1;
for(k=j+1;k<=n;k++) { a*=k; }
for(k=1;k<=(n-j);k++) { a=a/k; }
c=a;
return(c);
```

}