CG Practical Soft Copy

Submitted To:

ABHISHEK SHRIVASTAVA SIR

CSE Department

NIT Raipur

Submitted By:

Name: Vivek Kumar

Branch: CSE

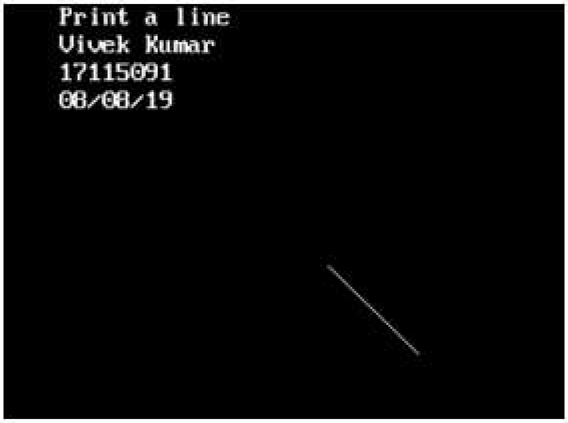
Semester: 5th

Roll N: 17115091

1) Program to Print a Line

Program code:

```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{       clrscr();

        int gd=DETECT,gm;

        initgraph(&gd,&gm,"..\\bgi");

        cout<<"Print a Line\nVivek Kumar \n17115091\n08/08/19";

        line(200,200,400,200);

        getch();

        closegraph();

}
```

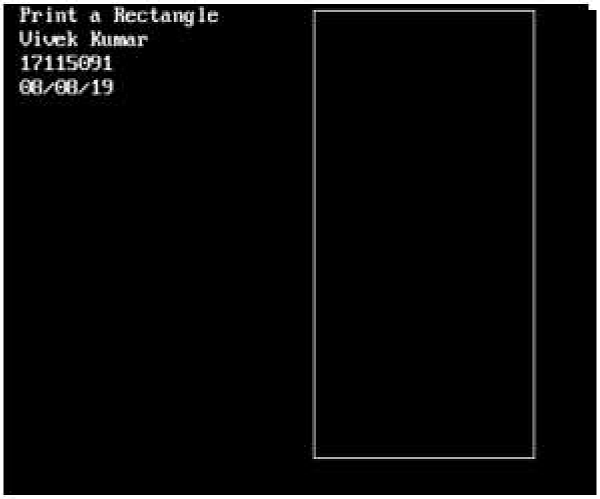


```
Print a line
Vivek Kumar
17115091
08/08/19
```

2)WAP to Print a Rectangle.

Program Code:

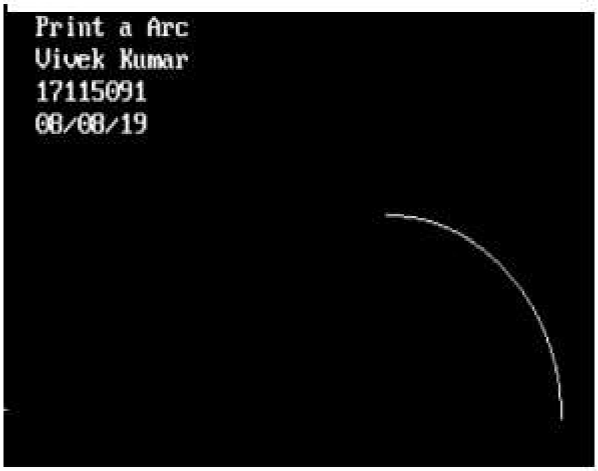
```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{       clrscr();

            int gd=DETECT,gm;

            initgraph(&gd,&gm,"..\\bgi");

            cout<<"Print a Rectangle\nVivek Kumar\n17115091\n08/08/19";

            rectangle(200,4,350,300);

            getch();

            closegraph();

}
```



```
Print a Rectangle
Vivek Kumar
17115091
08/08/19
```

3) Print a Arc

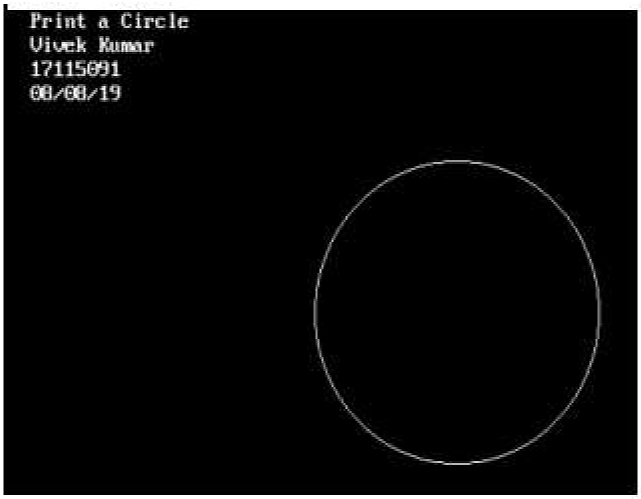
```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{       clrscr();

            int gd=DETECT,gm;

            initgraph(&gd,&gm,"..\\bgi");

            cout<<"Print a Arc\nVivek Kumar\n17115091\n08/08/19";

            arc(300,200,0,90,100);

            getch();

            closegraph();

}
```



```
Print a Arc
Vivek Kumar
17115091
08/08/19
```

4)WAP Print a Circle.

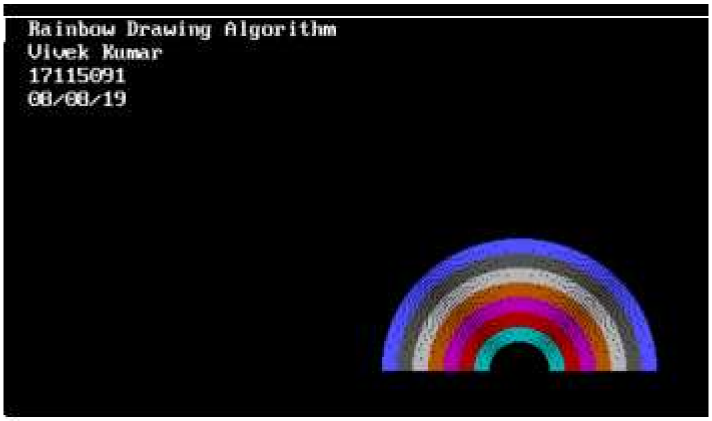
```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{       clrscr();

             int gd=DETECT,gm;

             initgraph(&gd,&gm,"..\\bgi");

             cout<<"Print a Circle\nVivek Kumar \n17115091\n08/08/19";

             circle(350,350,100);

             getch();

             closegraph();

}
```



```
Print a Circle
Vivek Kumar
17115091
08/08/19
```

5) Rainbow Drawing Algorithm

```
#include<graphics.h>

#include<constream.h>

#include<math.h>

#include<dos.h>

void main()

{     clrscr();

            int x,y;

            int i,gd=DETECT,gm;

            initgraph(&gd,&gm,"..\\bgi");

            cout<<"Rainbow Drawing Algorithm\nVivek Kumar\n17115091\n08/08/19";

            x=getmaxx()/2;

            y=getmaxy()/2;

            for(i=30;i<100;i++)

            {

                        delay(10);

                        setcolor(i/10);

                        arc(x,y,0,180,i-10);

            }

            getch();

            closegraph();

}
```

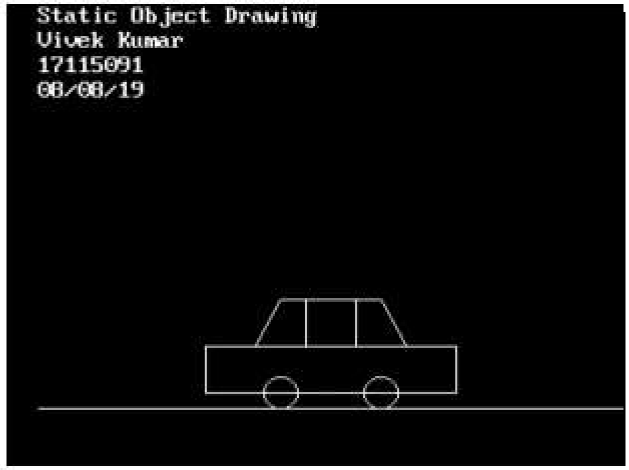


```
Rainbow Drawing Algorithm
Vivek Kumar
17115091
08/08/19
```

6) Static Object Drawing

```
#include<graphics.h>

#include<constream.h>

#include<math.h>

#include<dos.h>

void main()

{     clrscr();

            int x,y;

            int i,gd=DETECT,gm;

            initgraph(&gd,&gm,"..\\bgi");

            cout<<"Static Object Drawing\nVivek Kumar\n17115091\n08/08/19";

            rectangle(100,220,250,250);

            rectangle(160,190,190,220);

            line(145,190,205,190);

            line(145,190,130,220);

            line(205,190,220,220);

            circle(145,250,10);

            circle(205,250,10);

            line(0,260,650,260);
```

```
        getch();
        closegraph();



}
```

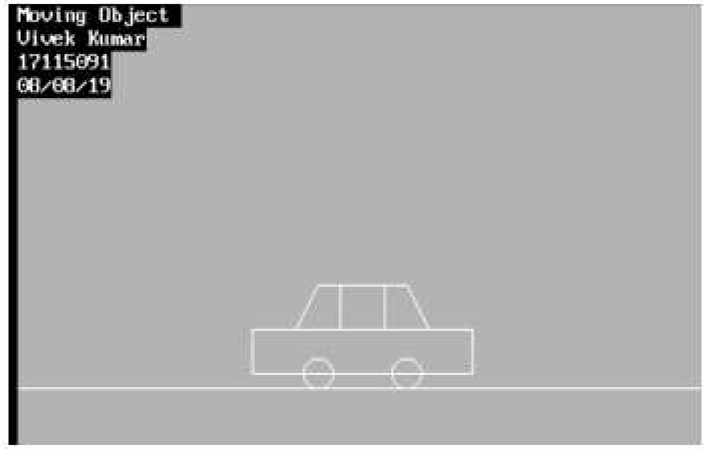


```
Static Object Drawing
Vivek Kumar
17115091
08/08/19
```

7) Moving Object

```
#include<graphics.h>
#include<constream.h>
#include<math.h>
#include<dos.h>
void main()
{     clrscr();
        int x,y;
        int i,gd=DETECT,gm;
        initgraph(&gd,&gm,"..\\bgi");
        for( i=10;i<=300;i+=5)
    {      clrscr();
```

```
        cout<<"Moving Object \nVivek Kumar\n17115091\n08/08/19";
        line(0,260,650,260);
        rectangle(100+i,220,250+i,250);
        rectangle(160+i,190,190+i,220);
        line(145+i,190,205+i,190);
        line(145+i,190,130+i,220);
        line(205+i,190,220+i,220);
        circle(145+i,250,10);
        circle(205+i,250,10);
        delay(80);
   }
     getch();
     closegraph();


}
```



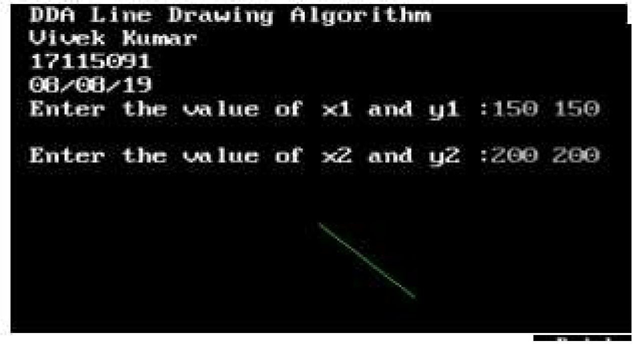
```
Moving Object
Vivek Kumar
17115091
08/08/19
```

8) DDA Line Drawing Algorithm

```
#include<graphics.h>
#include<constream.h>
#include<math.h>
```

```
#include<dos.h>
void main()
{	clrscr();
		float x,y,x1,y1,x2,y2,dx,dy,step;
		int i,gd=DETECT,gm;
		initgraph(&gd,&gm,"..\\bgi");
		cout<<"DDA Line Drawing Algorithm\nVivek Kumar\n17115091\n08/08/19";
		cout<<"\nEnter the value of x1 and y1 :";
		cin>>x1>>y1;
		cout<<"\nEnter the value of x2 and y2 :";
		cin>>x2>>y2;
		dx=abs(x2-x1);
		dy=abs(y2-y1);
		if(dx>=dy)
			step=dx;
		else
			step=dy;
		dx=dx/step;
		dy=dy/step;
		x=x1;
		y=y1;
		i=1;
		while(i<=step)
		{
			putpixel(x,y,10);
			x=x+dx;
			y=y+dy;
			i=i+1;
			delay(10);
```

```
        }
        getch();
        closegraph();




}
```



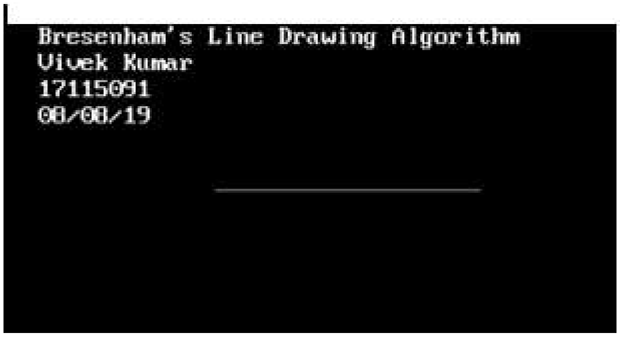
```
DDA Line Drawing Algorithm
Vivek Kumar
17115091
08/08/19
Enter the value of x1 and y1 :150 150

Enter the value of x2 and y2 :200 200
```

9) Bresenham's Line Drawing Algorithm

```
#include<graphics.h>
#include<constream.h>
#include<math.h>
#include<dos.h>
void main()
{    clrscr();
        float x,y,x2,y2,x1,y1,dx,dy,p;
        int i,gd=DETECT,gm;
        initgraph(&gd,&gm,"..\\bgi");
        cout<<"Bresenham's Line Drawing Algorithm\nVivek Kumar\n17115091\n08/08/19";
        //cout<<"\nEnter the value of x1 and y1 :";
        //cin>>x0>>y0;
```

```
//cout<<"\nEnter the value of x2 and y2 :";
//cin>>x1>>y1;
x1=100;y1=100;x2=250;y2=200;
dx=x1-x0;
dy=y1-y0;
x=x0;
y=y0;
p=2*dy-dx;
while(x<x1)
{
        if(p>=0)
        {
                putpixel(x,y,7);
                delay(10);
                y+=1;
                p=p+2*dy-2*dx;
        }
        else
        {
                putpixel(x,y,7);
                delay(10);
                p+=2*dy;
        }
        x=x+1;

}
getch();
closegraph();
```

```
}
```

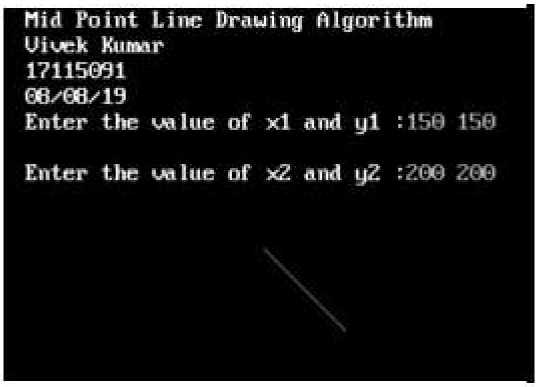


```
Bresenham's Line Drawing Algorithm
Vivek Kumar
17115091
08/08/19
```

10) Mid Point Line Drawing Algorithm

```
#include<graphics.h>
#include<constream.h>
#include<math.h>
#include<dos.h>
void main()
{     clrscr();
            float x,y,x1,y1,x2,y2,dx,dy,step;
            int i,gd=DETECT,gm;
            initgraph(&gd,&gm,"..\\bgi");
            cout<<"Mid Point Line Drawing Algorithm\nVivek Kumar\n17115091\n08/08/19";
            cout<<"\nEnter the value of x1 and y1 :";
            cin>>x1>>y1;
            cout<<"\nEnter the value of x2 and y2 :";
            cin>>x2>>y2;
            dx=x2-x1;
            dy=y2-y1;
            int d=dy-(dx/2);
```

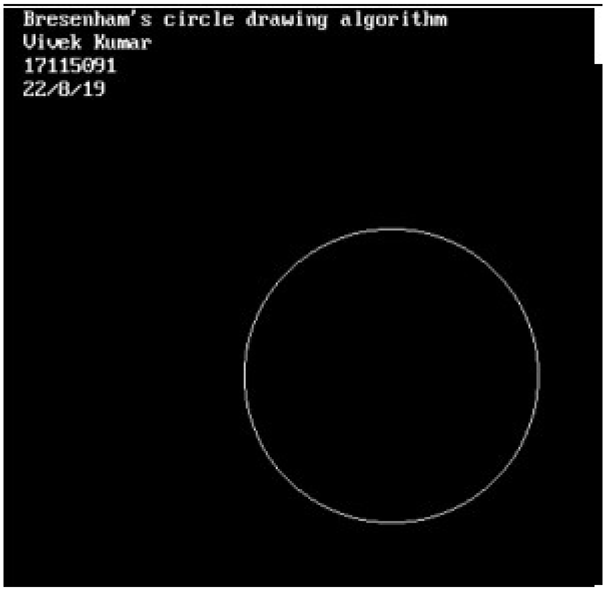
```
     x=x1,y=y1;
    putpixel(x,y,7);
    while(x<=x2)
    {
        x++;
        if(d<0)
            d=d+dy;
        else
        {
            d+=(dy-dx);
            y++;
        }
        putpixel(x,y,7);
        delay(10);
    }
    getch();
    closegraph();



}
```



```
Mid Point Line Drawing Algorithm
Vivek Kumar
17115091
08/08/19
Enter the value of x1 and y1 :150 150

Enter the value of x2 and y2 :200 200
```

11) Bresenham's circle drawing algorithm.

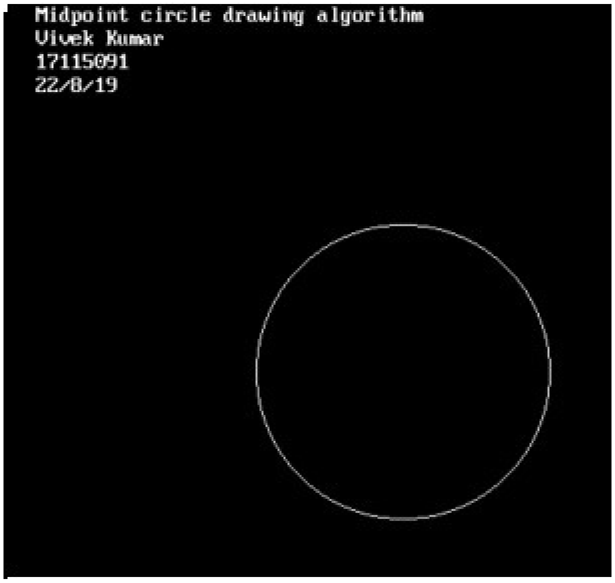
```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

    int gd=DETECT,gm;

    initgraph(&gd,&gm,"..\\BGI");

        cout<<"Bresenham's circle drawing algorithm.\nVivek Kumar\n17115091\n22/8/19";

        circle(250,250,100);

        getch();

        closegraph();

}
```



```
Bresenham's circle drawing algorithm
Vivek Kumar
17115091
22/8/19
```

12) Midpoint circle drawing algorithm

```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

    int gd=DETECT,gm;

    initgraph(&gd,&gm,"..\\BGI");

        cout<<"Midpoint circle drawing algorithm\nVivek Kumar\n17115091\n22/8/19";

        circle(250,250,100);

        getch();

        closegraph();

}
```



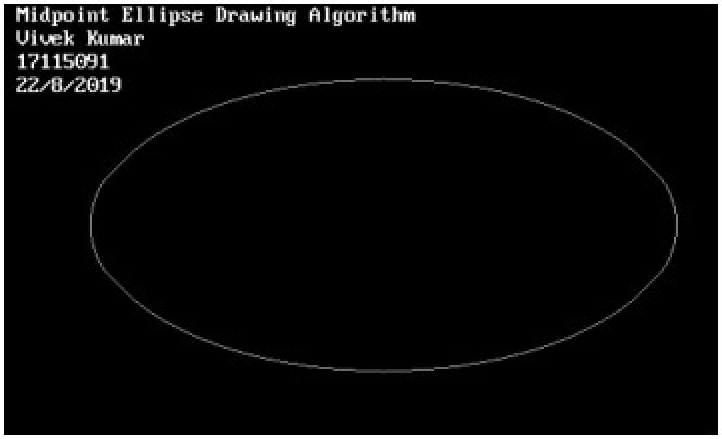
```
Midpoint circle drawing algorithm
Vivek Kumar
17115091
22/8/19
```

13) Ellipse using Mid point

```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>


void main()

{	clrscr();

		int gd=DETECT,gm;

		initgraph(&gd,&gm,"..\\bgi");

		clrscr();

		cout<<"Ellipse using Mid point\nVivek Kumar\n17115091\n22/8/19";

		int xc,yc,rx,ry;

		cout<<"\n Enter xc: ";

		cin>>xc;

		cout<<"\n Enter yc: ";

		cin>>yc;

		cout<<"\n Enter rx: ";

		cin>>rx;

		cout<<"\n Enter ry: ";

		cin>>ry;

		int x,y,p;

		x=0;

		y=ry;

		p=(ry*ry)-(rx*rx*ry)+((rx*rx)/4);

		while((2*x*ry*ry)<(2*y*rx*rx))

		{

			putpixel(xc+x,yc-y,WHITE);

			putpixel(xc-x,yc+y,WHITE);
```

```
        putpixel(xc+x,yc+y,WHITE);
        putpixel(xc-x,yc-y,WHITE);
        if(p<0)
        {
                x+=1;
                p+=(2*ry*ry*x)+(ry*ry);
        }
        else
        {
                x+=1;
                y-=1;
                p+=(2*ry*ry*x+ry*ry)-(2*rx*rx*y);
        }
}
p=((float)x+0.5)*((float)x+0.5)*ry*ry+(y-1)*(y-1)*rx*rx-rx*rx*ry*ry;
while(y>=0)
{
        putpixel(xc+x,yc-y,WHITE);
        putpixel(xc-x,yc+y,WHITE);
        putpixel(xc+x,yc+y,WHITE);
        putpixel(xc-x,yc-y,WHITE);
        if(p>0)
        {
                y-=1;
                p-=(2*rx*rx*y)+(rx*rx);
        }
        else
        {
                x+=1;
```

```
            y-=1;

            p+=(2*ry*ry*x)-(2*rx*rx*y)-(rx*rx);

        }

    }

    getch();

    clrscr();

    closegraph();

}
```

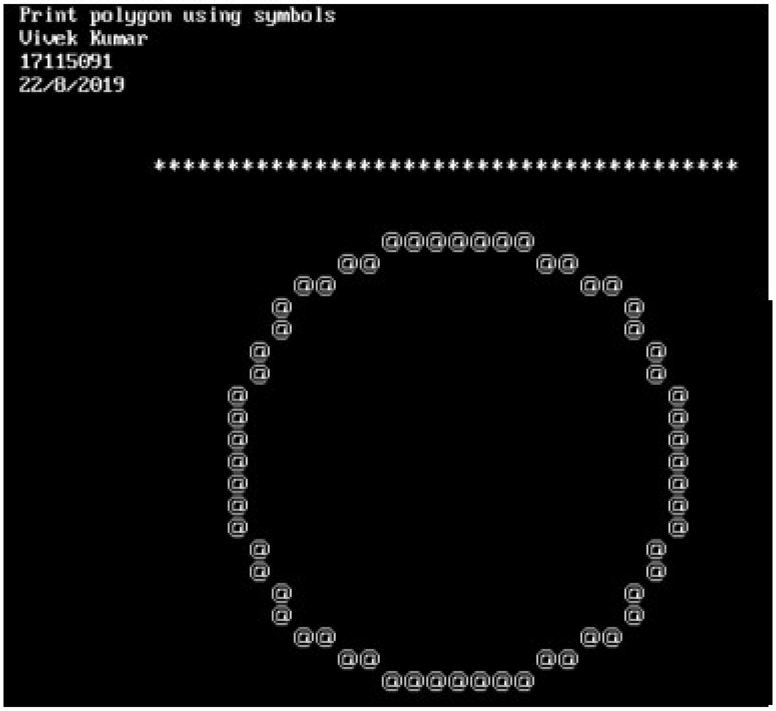


```
Midpoint Ellipse Drawing Algorithm
Vivek Kumar
17115091
22/8/2019
```

14) Print polygon using symbols

```
#include <stdio.h>

#include <iostream.h>

#include <conio.h>

#include <graphics.h>


void draw(int xc,int yc,int x,int y)

{

	outtextxy(xc+x,yc+y,"@");

	outtextxy(xc-x,yc+y,"@");

	outtextxy(xc-x,yc-y,"@");

	outtextxy(xc+x,yc-y,"@");

	outtextxy(xc+y,yc+x,"@");

	outtextxy(xc+y,yc-x,"@");

	outtextxy(xc-y,yc-x,"@");

	outtextxy(xc-y,yc+x,"@");

}


int main()

{

	int gd=DETECT,gm;

	initgraph(&gd,&gm,"..\\bgi");

	settextstyle(1,0,1);

	int a=100,b=100;

	cout<<" Print polygon using symbols\n Vivek Kumar\n 17115091\n 22/8/2019";

	for(;a<500;a+=10)

	{

		outtextxy(a,b,"*");
```

```
        }
        int xc=300,yc=300,r=150;
        int x=0,y=r;
        int d=3-2*r;
        draw(xc,yc,x,y);
        while(x<=y)
        {
                x+=15;
                if(d>0)
                {
                        y-=15;
                        d=d+4*(x-y)+10;
                }
                else
                {
                        d=d+4*x+6;
                }
                draw(xc,yc,x,y);
        }
        getch();
        closegraph();
        return 0;
}
```

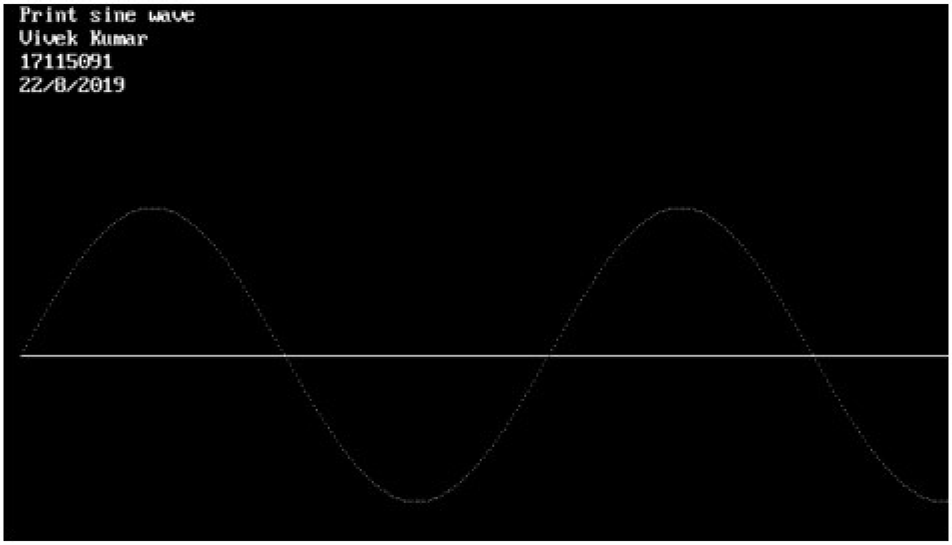


```
Print polygon using symbols
Vivek Kumar
17115091
22/8/2019


            ****************************************


                         @@@@@@@
                       @@       @@
                     @@           @@
                    @               @
                    @               @
                   @                 @
                   @                 @
                  @                   @
                  @                   @
                  @                   @
                  @                   @
                  @                   @
                  @                   @
                  @                   @
                   @                 @
                   @                 @
                    @               @
                    @               @
                     @@           @@
                       @@       @@
                         @@@@@@@
```

15) Print Sine and Cos wave

```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>


void main()

{       clrscr();

            int gd=DETECT,gm;

            initgraph(&gd,&gm,"..\\bgi");

            clrscr();

            cout<<"Print Sine and Cos wave\nVivek Kumar\n17115091\n22/8/19";

            line(100,160,500,160);
```

```
    line(100,50,100,350);
    arc(140,160,0,180,40);
    arc(220,160,-180,0,40);
    arc(300,160,0,180,40);
    arc(380,160,180,0,40);
    line(100,300,500,300);
    arc(100,300,0,90,40);
    arc(180,300,-180,0,40);
    arc(260,300,0,180,40);
    arc(340,300,180,0,40);
    arc(420,300,90,180,40);
    //circle(100,175,175);
    getch();
    clrscr();
    closegraph();
}
```



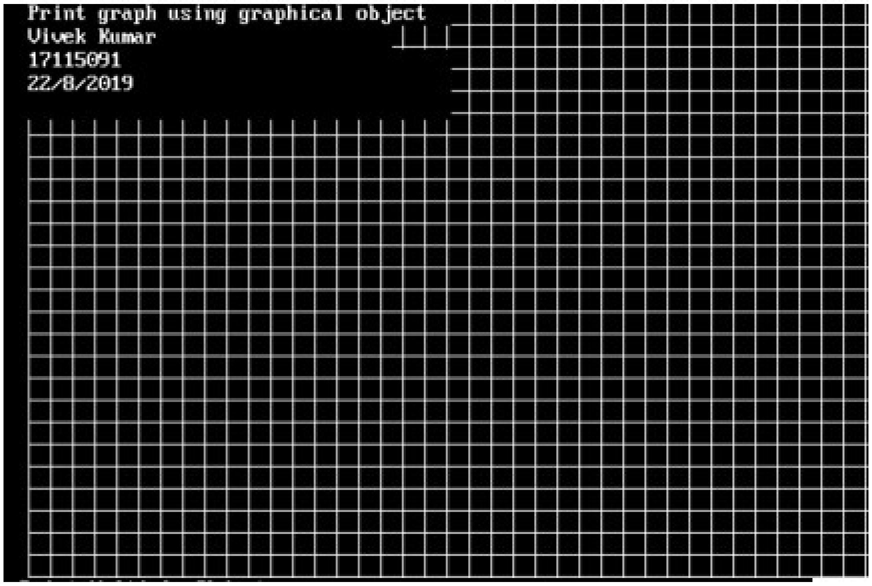
```
Print sine wave
Vivek Kumar
17115091
22/8/2019
```

16) Print graph using graphical object

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

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

int main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"..\\bgi");
    int x,y=0,xm,ym;
    xm=getmaxx();
    ym=getmaxy();

    for(x=0;x<=xm;x+=15,y+=15)
    {
        line(0,y,xm,y);
        line(y,0,y,ym);
        //y+=15;
    }
    /*
    for(x=0;x<=xm;x+=15)
    {
        circle(xm/2,ym/2,x);
    }
    */
    cout<<"Print graph using graphical object  ";
    cout<<"\nVivek Kumar                     ";
    cout<<"\n17115091                               ";
    cout<<"\n22/8/2019                               ";
    cout<<"\n                                         ";
    getch();
```

```
        closegraph();
        return 0;

}
```

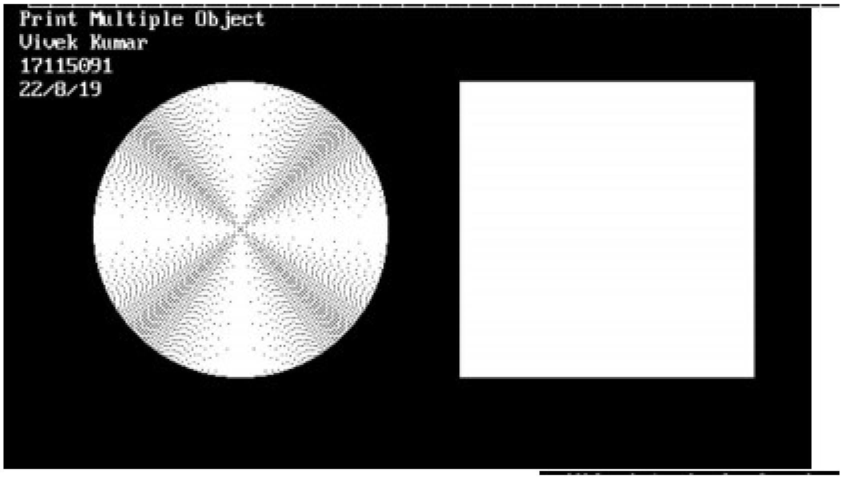


```
Print graph using graphical object
Vivek Kumar
17115091
22/8/2019
```

17) Print Multiple Object

```
#include<iostream>
#include<graphics.h>
using namespace std;
int main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"..\\bgi");
        cout<<"Print Multiple Object\nVivek Kumar\n17115091\n22/8/19";
        line(100,4,100,350);
        arc(100,175,-90,90,175);
```

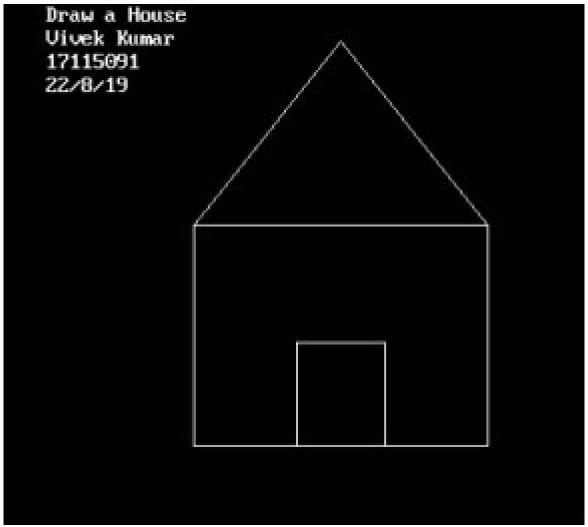
```
         line(400,4,400,350);
        arc(400,100,-90,90,95);


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



```
Print Multiple Object
Vivek Kumar
17115091
22/8/19
```

18) Draw a House

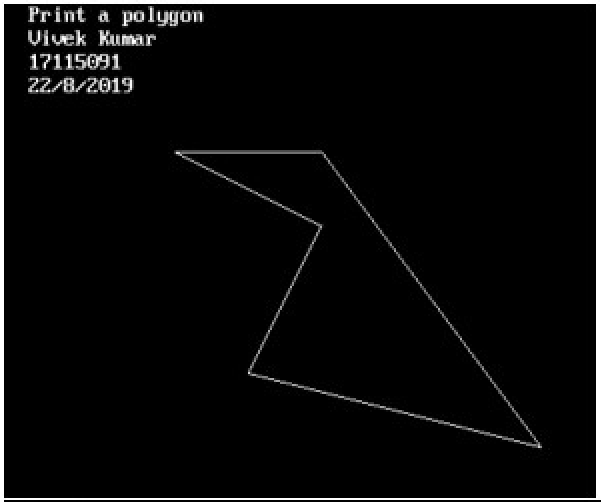
```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>


void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"..\\bgI");
        cout<<"Draw a House\nVivek Kumar\n17115091\n22/8/19";
        rectangle(100,150,300,300);
        line(200,25,100,150);
        line(200,25,300,150);

        rectangle(170,230,230,300);

        getch();

        closegraph();

}
```



```
Draw a House
Vivek Kumar
17115091
22/8/19
```

19) Print a polygon

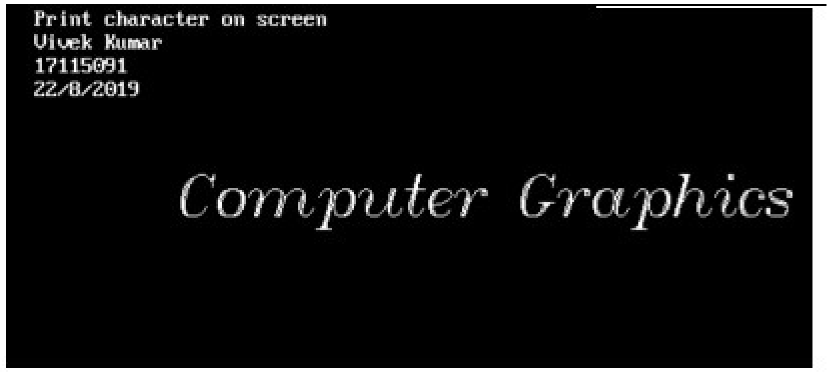
```
#include <iostream.h>

#include <conio.h>

#include <graphics.h>


int main()

{

        int gd=DETECT,gm;

        initgraph(&gd,&gm,"..\\bgi");

        cout<<"Print a polygon\nVivek Kumar\n17115091\n22/8/2019";

        int a[]={100,100,200,100,350,300,150,250,200,150,100,100};

        drawpoly(6,a);
        getch();
        closegraph();
        return 0;
}
```



```
Print a polygon
Vivek Kumar
17115091
22/8/2019
```

20) Print character on screen

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

int main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"..\\bgi");
        cout<<"Print character on screen\nVivek Kumar\n17115091\n22/8/2019";
        settextstyle(7,0,5);
        outtextxy(100,100,"Computer Graphics");
        getch();
        closegraph();
        return 0;
}
```

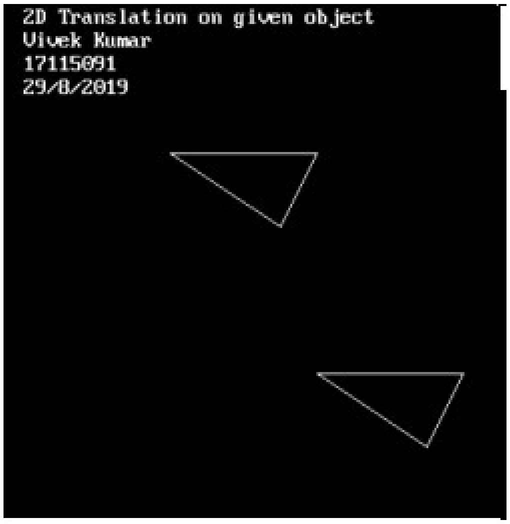


```
Print character on screen
Vivek Kumar
17115091
22/8/2019

Computer Graphics
```

21) 2D Translation on given object

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

int main()
{
        int gd=DETECT,gm;
        int tx=100,ty=150;
        int a[]={100,100,200,100,175,150,100,100};
        initgraph(&gd,&gm,"..\\bgi");
        cout<<"2D Translation on given object\nVivek Kumar\n17115091\n29/8/2019";
        drawpoly(4,a);
        for (int i=0;i<sizeof(a);i+=2)
        {
                a[i]+=tx;
                a[i+1]+=ty;
        }
        drawpoly(4,a);
        getch();
```

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

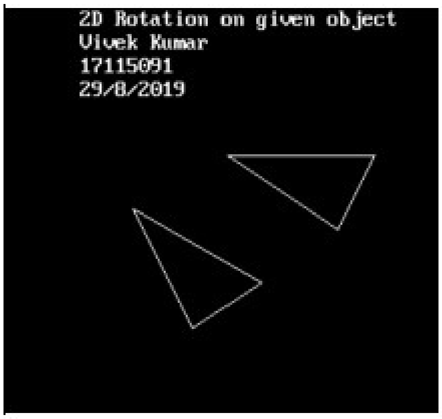


```
2D Translation on given object
Vivek Kumar
17115091
29/8/2019
```

22) 2D Rotation on given object

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

int main()
{
            int gd=DETECT,gm;
            initgraph(&gd,&gm,"..\\bgi");
            cout<<"2D Rotation on given object\nVivek Kumar\n17115091\n29/8/2019";
            int a[]={100,100,200,100,175,150,100,100},i,t;
```

```
        float ang=30*3.141592/180;

        drawpoly(4,a);

        for(i=0;i<sizeof(a);i+=2)

        {

                t=a[i];

                a[i]=a[i]*cos(ang)-a[i+1]*sin(ang);

                a[i+1]=t*sin(ang)+a[i+1]*cos(ang);

        }

        drawpoly(4,a);

        getch();

        getch();

        closegraph();

        return (0);

}
```

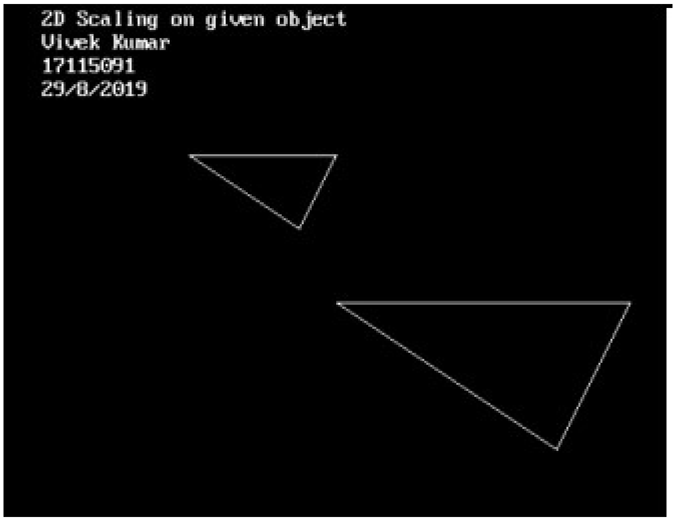


```
2D Rotation on given object
Vivek Kumar
17115091
29/8/2019
```

23) 2D Scaling on given object

```
#include <graphics.h>

#include <stdio.h>

#include <conio.h>

#include <iostream.h>


int main()
```

```
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"..\\bgi");
        cout<<"2D Scaling on given object\nVivek Kumar\n17115091\n29/8/2019";
        int a[]={100,100,200,100,175,150,100,100};
        int sx=2,sy=2,i=0;
        drawpoly(4,a);
        for(;i<8;i+=2)
        {
                a[i]*=sx;
                a[i+1]*=sy;
        }
        drawpoly(4,a);
        getch();
        getch();
        closegraph();
        return (0);
}
```

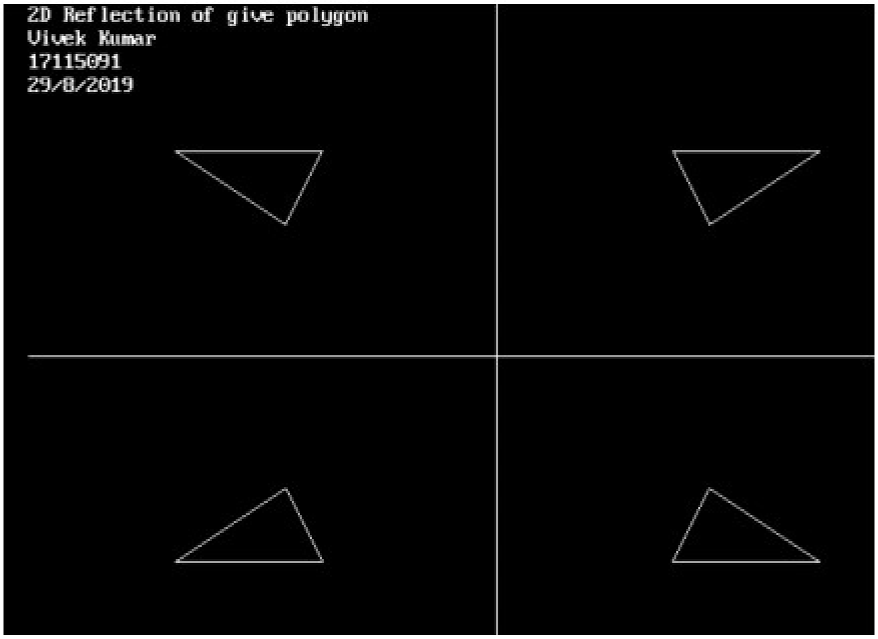


```
2D Scaling on given object
Vivek Kumar
17115091
29/8/2019
```

24) 2D Reflection of give polygon

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

int main()
{
	int gd=DETECT,gm;
	int a[8]={100,100,200,100,175,150,100,100};
	int b[8],c[8];
	int xm,ym,i;
	initgraph(&gd,&gm,"..\\bgi");
	xm=getmaxx();
	ym=getmaxy();
	line(0,ym/2,xm,ym/2);
	line(xm/2,0,xm/2,ym);
	drawpoly(4,a);
	cout<<"2D Reflection of give polygon\nVivek Kumar\n17115091\n29/8/2019";
	for(i=0;i<sizeof(a)/2;i+=2)
	{
		b[i]=a[i];
		b[i+1]=ym-a[i+1];
	}
	drawpoly(4,b);
	for(i=0;i<sizeof(a)/2;i+=2)
	{
		c[i]=xm-a[i];
		c[i+1]=a[i+1];
	}
```

```
        drawpoly(4,c);
        for(i=0;i<sizeof(a)/2;i+=2)
        {
                c[i]=xm-a[i];
                c[i+1]=ym-a[i+1];
        }
        drawpoly(4,c);
        getch();
        getch();
        closegraph();
        return 0;
}
```

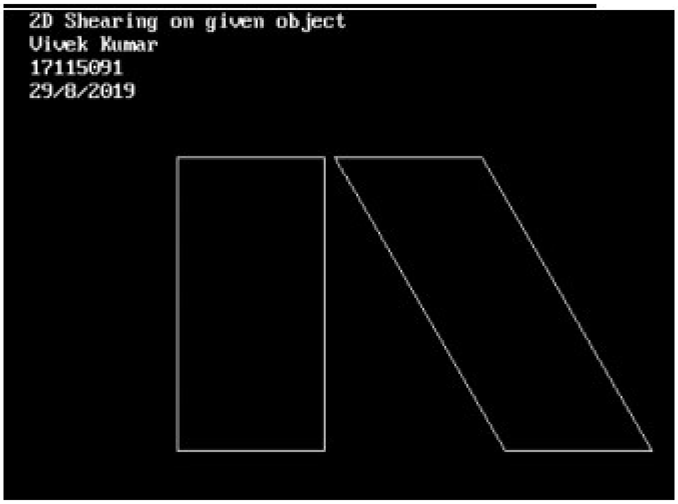


```
2D Reflection of give polygon
Vivek Kumar
17115091
29/8/2019
```

25) 2D Shearing on given object

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

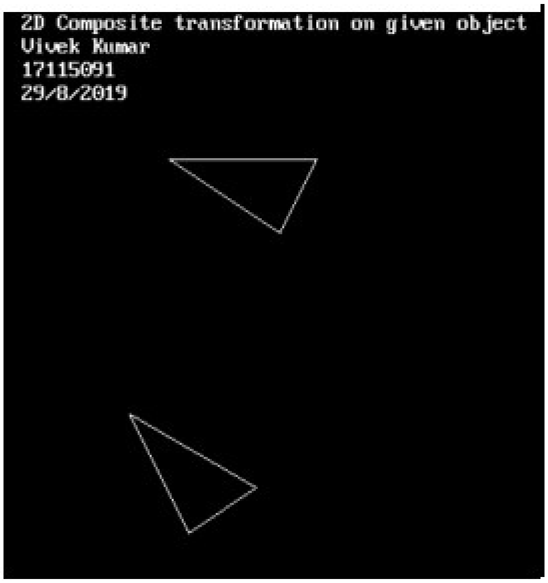
```
#include <math.h>
#include <dos.h>

int main()
{
        int gd=DETECT,gm;
        int a[]={100,100,200,100,200,300,100,300,100,100},i;
        float n=30*3.141592/180;
        initgraph(&gd,&gm,"..\\bgi");
        drawpoly(5,a);
        cout<<"2D Shearing on given object\nVivek Kumar\n17115091\n29/8/2019";
        for(i=0;i<sizeof(a);i+=2)
        {
                a[i]=50+a[i]+a[i+1]*tan(n);
        }
        drawpoly(5,a);
        getch();
        getch();
        closegraph();
        return (0);
}
```



```
2D Shearing on given object
Vivek Kumar
17115091
29/8/2019
```

26) 2D Composite transformation on given object

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

int main()
{
        int gd=DETECT,gm;
        int tx=100,ty=100,i,t;
        float n=30*3.141592/180;
        int a[8]={100,100,200,100,175,150,100,100};
        initgraph(&gd,&gm,"..\\bgi");
        cout<<"2D Composite transformation on given object\nVivek Kumar\n17115091\n29/8/2019";
        drawpoly(4,a);
        for (i=0;i<sizeof(a)/2;i+=2)
        {
                a[i]+=tx;
                a[i+1]+=ty;
        }
        for (i=0;i<sizeof(a)/2;i+=2)
        {
                t=a[i];
                a[i]=a[i]*cos(n)-a[i+1]*sin(n);
                a[i+1]=t*sin(n)+a[i+1]*cos(n);
        }
        drawpoly(4,a);
        getch();
        getch();
        closegraph();
        return (0);
}
```

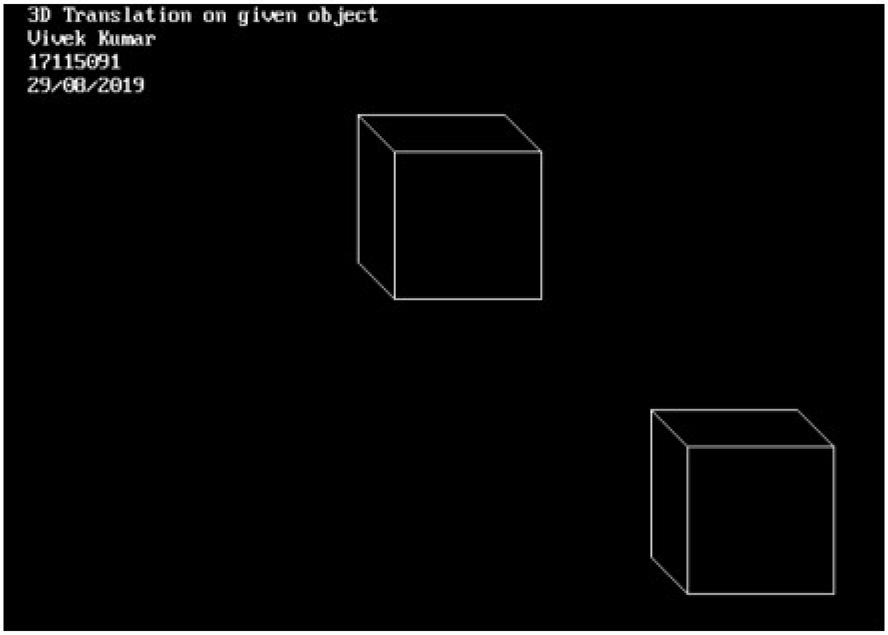


```
2D Composite transformation on given object
Vivek Kumar
17115091
29/8/2019
```

27) WAP to perform 3d translation in object

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

```
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
getch();
cleardevice();
cout<<"WAP to perform 3d translation in object"<<endl;
cout<<"Vivek Kumar"<<endl;
cout<<"17115091"<<endl;
cout<<"29/08/2019"<<endl;
line(midx,0,midx,maxy);
line(0,midy,maxx,midy);
}
void main()
{
int x,y,z,o,x1,x2,y1,y2;
int gd=DETECT,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"..\\bgi");
setfillstyle(0,getmaxcolor());
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;

axis();

bar3d(midx-30,midy+100,midx-40,midy+90,10,10);
printf("Enter translation factor");
scanf("%d%d",&x,&y);
//axis();
printf("After translation:");
bar3d(midx+x+150,midy-(y+100),midx+x+140,midy-(y+90),10,10);
getch();
closegraph();
}
```

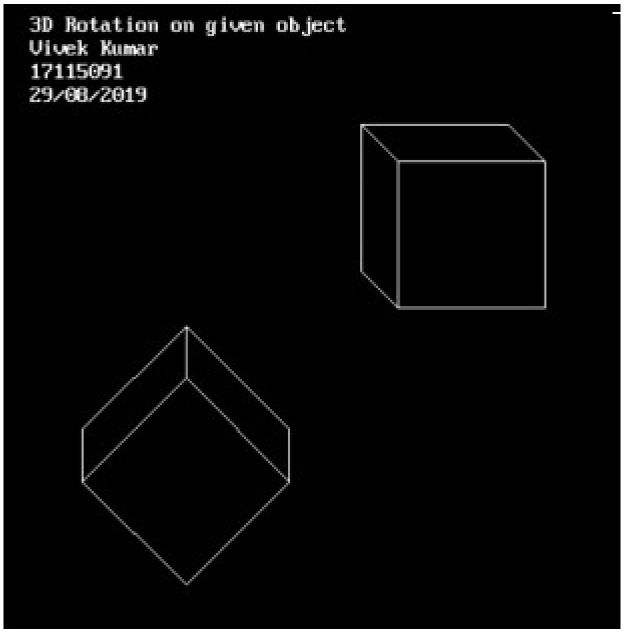


```
3D Translation on given object
Vivek Kumar
17115091
29/08/2019
```

28) WAP to perform 3d rotation in object

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

```
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
getch();
cleardevice();
cout<<"WAP to perform 3d rotation in object"<<endl;
cout<<"Vivek Kumar"<<endl;
cout<<"17115091"<<endl;
cout<<"29/08/2019"<<endl;
line(midx,0,midx,maxy);
line(0,midy,maxx,midy);
}
void main()
{
int x,y,z,o,x1,x2,y1,y2;
int gd=DETECT,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"..\\bgi");
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;
axis();
bar3d(midx+50,midy-100,midx+60,midy-90,100,20);
printf("Enter rotating angle");
scanf("%d",&o);
 x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);
 y1=50*sin(o*3.14/180)+100*cos(o*3.14/180);
 x2=60*cos(o*3.14/180)-90*sin(o*3.14/180);
 y2=60*sin(o*3.14/180)+90*cos(o*3.14/180);
 axis();
printf("After rotation about z axis");
bar3d(midx+x1,midy-y1,midx+x2,midy-y2,100,20);
axis();
printf("After rotation  about x axis");
bar3d(midx+50,midy-x1,midx+60,midy-x2,100,20);
axis();
printf("After rotation about yaxis");
bar3d(midx+x1,midy-100,midx+x2,midy-90,100,4);
getch();
closegraph();
}
```

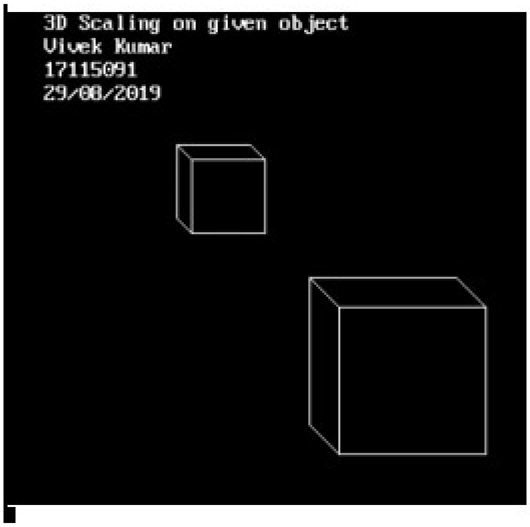


```
3D Rotation on given object
Vivek Kumar
17115091
29/08/2019
```

29) WAP to perform 3d scaling in object

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

```
#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int maxx,maxy,midx,midy;

void axis()

{

getch();

cleardevice();

cout<<"WAP to perform 3d scaling in object"<<endl;

cout<<"Vivek Kumar"<<endl;

cout<<"17115091"<<endl;

cout<<"29/08/2019"<<endl;

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

}

void main()

{

int x,y,z,o,x1,x2,y1,y2;

int gd=DETECT,gm;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"..\\bgi");

cout<<"WAP to perform 3d scaling in object"<<endl;

cout<<"NEHA AGRAWAL"<<endl;

cout<<"16115049"<<endl;

cout<<"27/08/2019"<<endl;

//setfillstyle(0,getmaxcolor());

maxx=getmaxx();

maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
printf("Enter scaling factors");
scanf("%d%d%d", &x,&y,&z);
//axis();
printf("After scaling");
bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);
//axis();
getch();
closegraph();
}
```



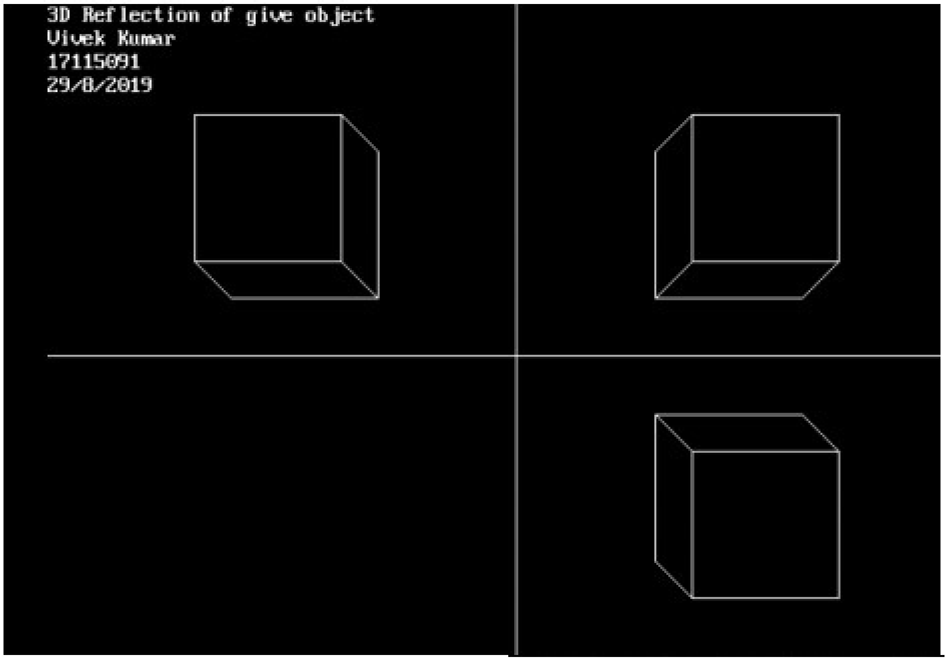
```
3D Scaling on given object
Vivek Kumar
17115091
29/08/2019
```

30) WAP to perform 3d reflection in object

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

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

#define ORG -50

double face1[5][2] = {
    { 250, 125 },
    { 350, 125 },
    { 350, 225 },
    { 250, 225 },
    { 250, 125 }
        };

double face2[5][2] = {
    { 250+ORG, 125-ORG },
    { 350+ORG, 125-ORG },
    { 350+ORG, 225-ORG },
    { 250+ORG, 225-ORG },
    { 250+ORG, 125-ORG }
        };

double angle = 5.0 * M_PI / 180;
double midx1, midy1, midx2, midy2;

void rotate (void)
{
   int i;
   for (i=0; i<5; i++)
   {
double xnew, ynew;

xnew = midx1 + (face1[i][0] - midx1) * cos (angle) -
      (face1[i][1] - midy1) * sin (angle);
ynew = midy1 + (face1[i][0] - midx1) * sin (angle) +
 (face1[i][1] - midy1) * cos (angle);

face1[i][0] = xnew;
face1[i][1] = ynew;

xnew = midx2 + (face2[i][0] - midx2) * cos (angle) -
 (face2[i][1] - midy2) * sin (angle);
ynew = midy2 + (face2[i][0] - midx2) * sin (angle) +
 (face2[i][1] - midy2) * cos (angle);

face2[i][0] = xnew;
face2[i][1] = ynew;
   }

   cleardevice();
   cout<<"WAP to perform 3d reflection in object"<<endl;
   cout<<"AMIT KUMAR"<<endl;
   cout<<"17115010"<<endl;
   cout<<"27/08/2019"<<endl;
   for (i=0; i<4; i++)
   {
```

```
 setcolor(7);
 line (face1[i][0], face1[i][1], face1[i+1][0], face1[i+1][1]);
 setcolor(8);
 line (face2[i][0], face2[i][1], face2[i+1][0], face2[i+1][1]);
 setcolor(9);
 line (face1[i][0], face1[i][1], face2[i][0], face2[i][1]);
    }

    delay (125);
}

void main()
{
    int gd = DETECT, gm;
    midx1 = (face1[0][0] + face1[1][0]) / 2.0;
    midy1 = (face1[1][1] + face1[2][1]) / 2.0;
    midx2 = (face2[0][0] + face2[1][0]) / 2.0;
    midy2 = (face2[1][1] + face2[2][1]) / 2.0;
    initgraph (&gd, &gm, "..\\bgi");
    cout<<"WAP to perform 3d reflection in object"<<endl;
    cout<<"Vivek Kumar"<<endl;
    cout<<"17115091"<<endl;
    cout<<"29/08/2019"<<endl;
    while (!kbhit())
 rotate();

    closegraph();
}
```



```
3D Ref lection of give object
Vivek Kumar
17115091
29/8/2019
```

31)3D shearing

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



 # define  f               0.3
 # define  projection_angle   45



 void show_screen( );


 void apply_x_shearing(int[5][3],constfloat,constfloat);
 void multiply_matrices(constfloat[4],constfloat[4][4],float[4]);
void draw_pyramid(constint [5][3]);
void get_projected_point(int&,int&,int&);

void Line(constint,constint,constint,constint);



int main( )
  {
    int driver=VGA;
    int mode=VGAHI;

    initgraph(&driver,&mode,"..\\Bgi");

    show_screen( );

    int pyramid[5][3]={
          {280,220,40},     //  base front left
          {360,220,40},     //  base front right
          {360,220,-40},    //  base back right
          {280,220,-40},    //  base back left
          {320,100,0}       //  top
        };

   setcolor(15);
   draw_pyramid(pyramid);

    setcolor(15);
    settextstyle(0,0,1);
   outtextxy(50,415,"*** Press any key to see the 3D Shearing along x-axis.");
```

```
    apply_x_shearing(pyramid,0.4,0.3);

    getch( );

    setcolor(10);
   draw_pyramid(pyramid);

    getch( );
    return 0;
  }

 /***********************************************************************///-------------
-----------  apply_x_shearing( )  -------------------
///***********************************************************************/void
apply_x_shearing(int edge_points[5][3],constfloat a,constfloat b)
  {
    for(int count=0;count<5;count++)
   {
      float matrix_a[4]={edge_points[count][0],edge_points[count][1],
                   edge_points[count][2],1};
      float matrix_b[4][4]={
              { 1,a,b,0 } ,
              { 0,1,0,0 } ,
              { 0,0,1,0 } ,
              { 0,0,0,1 }
            };

      float matrix_c[4]={0};
```

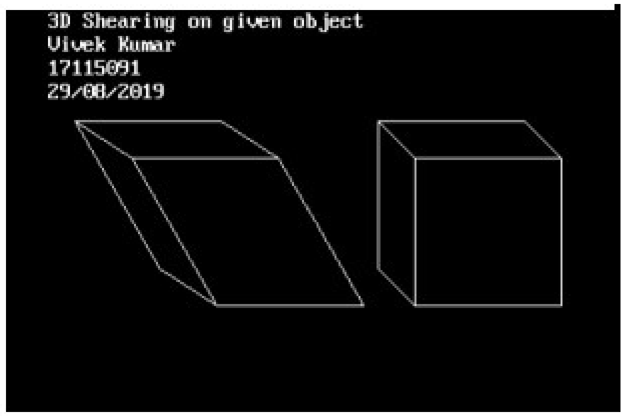
```
        multiply_matrices(matrix_a,matrix_b,matrix_c);


        edge_points[count][0]=(int)(matrix_c[0]+0.5);
        edge_points[count][1]=(int)(matrix_c[1]+0.5);
        edge_points[count][2]=(int)(matrix_c[2]+0.5);
      }
    }

 /**********************************************************************///----------------
------  multiply_matrices( )  ------------------------
///**********************************************************************/void
multiply_matrices(constfloat matrix_1[4],
              constfloat matrix_2[4][4],float matrix_3[4])
    {
      for(int count_1=0;count_1<4;count_1++)
      {
        for(int count_2=0;count_2<4;count_2++)
       matrix_3[count_1]+=
            (matrix_1[count_2]*matrix_2[count_2][count_1]);
      }
    }

 /**********************************************************************///----------------
----------  draw_pyramid( )  ------------------------
///**********************************************************************/void
draw_pyramid(constint points[5][3])
    {
      int edge_points[5][3];


      for(int i=0;i<5;i++)
      {
        edge_points[i][0]=points[i][0];
        edge_points[i][1]=points[i][1];
        edge_points[i][2]=points[i][2];

        get_projected_point(edge_points[i][0],
               edge_points[i][1],edge_points[i][2]);
      }

      Line(edge_points[0][0],edge_points[0][1],
               edge_points[1][0],edge_points[1][1]);
      Line(edge_points[1][0],edge_points[1][1],
               edge_points[2][0],edge_points[2][1]);
      Line(edge_points[2][0],edge_points[2][1],
               edge_points[3][0],edge_points[3][1]);
      Line(edge_points[3][0],edge_points[3][1],
               edge_points[0][0],edge_points[0][1]);

      Line(edge_points[0][0],edge_points[0][1],
               edge_points[4][0],edge_points[4][1]);
      Line(edge_points[1][0],edge_points[1][1],
               edge_points[4][0],edge_points[4][1]);
      Line(edge_points[2][0],edge_points[2][1],
               edge_points[4][0],edge_points[4][1]);
      Line(edge_points[3][0],edge_points[3][1],
               edge_points[4][0],edge_points[4][1]);
    }
```

```
 /******************************************************************************///--------------------
-----  get_projected_point( )  ----------------------
///****************************************************************************/void
get_projected_point(int& x,int& y,int& z)

   {

      float fcos0=(f*cos(projection_angle*(M_PI/180)));

      float fsin0=(f*sin(projection_angle*(M_PI/180)));


      float Par_v[4][4]={

             {1,0,0,0},

             {0,1,0,0},

             {fcos0,fsin0,0,0},

             {0,0,0,1}

           };


      float xy[4]={x,y,z,1};

      float new_xy[4]={0};


      multiply_matrices(xy,Par_v,new_xy);


      x=(int)(new_xy[0]+0.5);

      y=(int)(new_xy[1]+0.5);

      z=(int)(new_xy[2]+0.5);

   }


 /*******************************************************************************///---------------
-----------------  Line( )  ------------------------------
///*****************************************************************************/void
Line(constint x_1,constint y_1,constint x_2,constint y_2)

   {

      int color=getcolor( );
 int x1=x_1;
 int y1=y_1;

 int x2=x_2;
 int y2=y_2;

 if(x_1>x_2)
{
   x1=x_2;
   y1=y_2;

   x2=x_1;
   y2=y_1;
}

 int dx=abs(x2-x1);
 int dy=abs(y2-y1);
 int inc_dec=((y2>=y1)?1:-1);

 if(dx>dy)
{
   int two_dy=(2*dy);
   int two_dy_dx=(2*(dy-dx));
   int p=((2*dy)-dx);

   int x=x1;
   int y=y1;
```

```
  putpixel(x,y,color);

  while(x<x2)
 {
   x++;

   if(p<0)
     p+=two_dy;

   else
     {
    y+=inc_dec;
    p+=two_dy_dx;
     }

   putpixel(x,y,color);
 }
}

 else
{
  int two_dx=(2*dx);
  int two_dx_dy=(2*(dx-dy));
  int p=((2*dx)-dy);

  int x=x1;
  int y=y1;

  putpixel(x,y,color);
```

```
        while(y!=y2)
       {
          y+=inc_dec;

          if(p<0)
             p+=two_dx;

          else
             {
            x++;
            p+=two_dx_dy;
             }

          putpixel(x,y,color);
       }
     }
   }

 /************************************************************************///--------------
------------  show_screen( )  --------------------------
///************************************************************************/void
show_screen( )
   {
      setfillstyle(1,1);
    bar(210,26,420,38);

      settextstyle(0,0,1);
    setcolor(15);
```

```
outtextxy(5,5,"************************************************************************
*****");

      outtextxy(5,17,"*-
*************************************************************************-*");

      outtextxy(5,29,"*-----------------------                            ------------------------*");

      outtextxy(5,41,"*-
*************************************************************************-*");

      outtextxy(5,53,"*-
*************************************************************************-*");


    setcolor(11);

      outtextxy(218,29,"3D Shearing along x-axis");


    setcolor(15);


      for(int count=0;count<=30;count++)

         outtextxy(5,(65+(count*12)),"*-*                                                                       *-*");


      outtextxy(5,438,"*-
*************************************************************************-*");

      outtextxy(5,450,"*------------------------                           ------------------------*");

outtextxy(5,462,"**********************************************************************
*******");


    setcolor(12);

      outtextxy(229,450,"Press any Key to exit.");

   }
```



```
3D Shearing on given object
Vivek Kumar
17115091
29/08/2019
```

32)3D composite transformation

```
#include<iostream.h>

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

#include<math.h>

#define  f              0.3

#define  projection_angle   45

void trans();

void scale();

void rotate();

void show_screen( );

void apply_x_shearing(int[5][3],float,float);

void multiply_matrices(float[4],float[4][4],float[4]);

void draw_pyramid(int [5][3]);

void get_projected_point(int&,int&,int&);

void Line(int,int,int,int);

int maxx,maxy,midx,midy;

int main()

{
```

```
int ch;
int gd=DETECT,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"..\\bgi");
cout<<"WAP to perform 3D COMPOSITE translation of object:"<<endl;
cout<<"1.Translation \n2.Scaling\n 3.Rotation \n 4.Shearing \n5.Exit\n";
printf("enter your choice");
scanf("%d",&ch);
do
{
            switch(ch)
            {
                    case 1 :
                            trans();
                            getch();
                            break;
                     case 2 :
                            scale();
                            getch();
                            break;
                    case 3 :
                            rotate();
                            getch();                                    break;

                    case 4 :
                            show_screen( );
                            int pyramid[5][3]={
                                                {280,220,40},     //  base front left
                                                {360,220,40},     //  base front right
```

```
                                                      {360,220,-40},     //  base back right
                                                      {280,220,-40},     //  base back left
                                                      {320,100,0}        //  top

         };
   setcolor(15);
  draw_pyramid(pyramid);
   setcolor(15);
   settextstyle(0,0,1);
   outtextxy(50,455,"*** Press any key to see the 3D Shearing along x-axis.");
   apply_x_shearing(pyramid,0.4,0.3);
   getch( );
   setcolor(10);
   draw_pyramid(pyramid);
   getch( );
                                   break;
                            case 5:
                                   return 0;
                    }
                    printf("enter your choice");
                    scanf("%d",&ch);

        } while(ch<4);
        return 0;
}
void trans()
{

        int x,y,z,o,x1,x2,y1,y2;
```

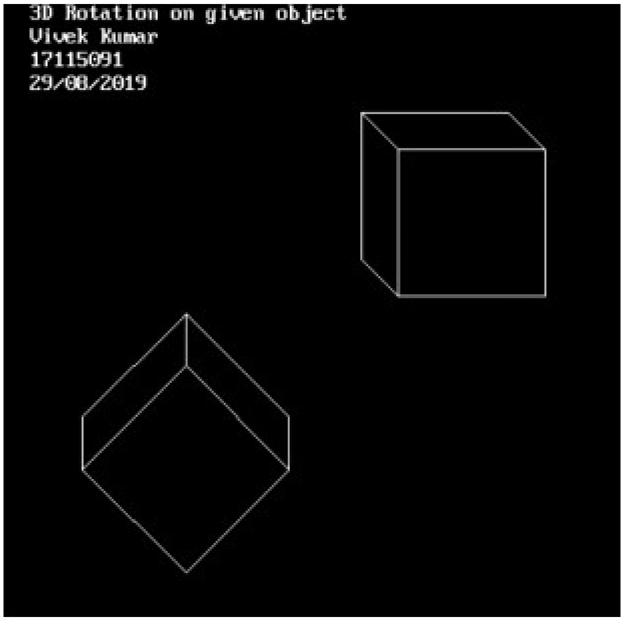
```
      maxx=getmaxx();
    maxy=getmaxy();
    midx=maxx/2;
    midy=maxy/2;
    bar3d(midx+50,midy-100,midx+60,midy-90,10,1);
    printf("Enter translation factor");
    scanf("%d%d",&x,&y);
    printf("After translation:");
    bar3d(midx+x+50,midy-(y+100),midx+x+60,midy-(y+90),10,1);
}
void scale()
{

    int x,y,z,o,x1,x2,y1,y2;
    maxx=getmaxx();
    maxy=getmaxy();
    midx=maxx/2;
    midy=maxy/2;
    bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
    printf("before translation\n");
    printf("Enter scaling factors\n");
    scanf("%d %d %d", &x,&y,&z);
    printf("After scaling\n");
    bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);
}
void rotate()
{
    int x,y,z,o,x1,x2,y1,y2;
    maxx=getmaxx();
```

```
            maxy=getmaxy();
            midx=maxx/2;
            midy=maxy/2;
                bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
                printf("Enter rotating angle");
                scanf("%d",&o);
                x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);
    //          y1=50*sin(o*3.14/180)+100*cos(o*3.14/180);
                x2=60*cos(o*3.14/180)-90*sin(o*3.14/180);
   //           y2=60*sin(o*3.14/180)+90*cos(o*3.14/180);
                printf("After rotation  about x axis");
                bar3d(midx+50,midy-x1,midx+60,midy-x2,5,1);
                printf("After rotation about yaxis");
                bar3d(midx+x1,midy-100,midx+x2,midy-90,5,1);
}
void    apply_x_shearing(int edge_points[5][3],float a,float b)
    {
     for(int count=0;count<5;count++)
     {
            float matrix_a[4]={edge_points[count][0],edge_points[count][1],
                                    edge_points[count][2],1};
            float matrix_b[4][4]={
                        { 1,a,b,0 } ,
                        { 0,1,0,0 } ,
                        { 0,0,1,0 } ,
                        { 0,0,0,1 }
                       };

            float matrix_c[4]={0};
            multiply_matrices(matrix_a,matrix_b,matrix_c);


            edge_points[count][0]=(int)(matrix_c[0]+0.5);

            edge_points[count][1]=(int)(matrix_c[1]+0.5);

            edge_points[count][2]=(int)(matrix_c[2]+0.5);

      }

    }
 /**********************************************************************///----------------
------  multiply_matrices( )  -----------------------
///**********************************************************************/void
multiply_matrices(constfloat matrix_1[4],

void  multiply_matrices(float matrix_1[4], float matrix_2[4][4], float matrix_3[4])

    {

      for(int count_1=0;count_1<4;count_1++)

      {

            for(int count_2=0;count_2<4;count_2++)

                        matrix_3[count_1]+=

                  (matrix_1[count_2]*matrix_2[count_2][count_1]);

      }

    }


 /**********************************************************************///----------------
----------  draw_pyramid( )  -------------------------
///**********************************************************************/void
draw_pyramid(constint points[5][3])

void draw_pyramid(int points[5][3])

 {

      int edge_points[5][3];


      for(int i=0;i<5;i++)
 {
         edge_points[i][0]=points[i][0];
         edge_points[i][1]=points[i][1];
         edge_points[i][2]=points[i][2];

         get_projected_point(edge_points[i][0],
                        edge_points[i][1],edge_points[i][2]);
 }

  Line(edge_points[0][0],edge_points[0][1],
                         edge_points[1][0],edge_points[1][1]);
  Line(edge_points[1][0],edge_points[1][1],
                         edge_points[2][0],edge_points[2][1]);
  Line(edge_points[2][0],edge_points[2][1],
                         edge_points[3][0],edge_points[3][1]);
  Line(edge_points[3][0],edge_points[3][1],
                         edge_points[0][0],edge_points[0][1]);

  Line(edge_points[0][0],edge_points[0][1],
                         edge_points[4][0],edge_points[4][1]);
  Line(edge_points[1][0],edge_points[1][1],
                         edge_points[4][0],edge_points[4][1]);
  Line(edge_points[2][0],edge_points[2][1],
                         edge_points[4][0],edge_points[4][1]);
  Line(edge_points[3][0],edge_points[3][1],
                         edge_points[4][0],edge_points[4][1]);
}
```

```
/**********************************************************************///----------------
----- get_projected_point( ) ----------------------
///**********************************************************************/void
get_projected_point(int& x,int& y,int& z)

void  get_projected_point(int &x,int &y,int &z)

{

      float fcos0=(f*cos(projection_angle*(M_PI/180)));

      float fsin0=(f*sin(projection_angle*(M_PI/180)));


      float Par_v[4][4]={

                        {1,0,0,0},

                        {0,1,0,0},

                        {fcos0,fsin0,0,0},

                        {0,0,0,1}

                 };


      float xy[4]={x,y,z,1};

      float new_xy[4]={0};


      multiply_matrices(xy,Par_v,new_xy);


      x=(int)(new_xy[0]+0.5);

      y=(int)(new_xy[1]+0.5);

      z=(int)(new_xy[2]+0.5);

}


/**********************************************************************///--------------
----------------- Line( ) ----------------------------
///**********************************************************************/void
Line(constint x_1,constint y_1,constint x_2,constint y_2)

void Line(int x_1,int y_1,int x_2,int y_2){
 int color=getcolor( );

 int x1=x_1;
 int y1=y_1;

 int x2=x_2;
 int y2=y_2;

 if(x_1>x_2)
{
         x1=x_2;
         y1=y_2;

         x2=x_1;
         y2=y_1;
}

 int dx=abs(x2-x1);
 int dy=abs(y2-y1);
 int inc_dec=((y2>=y1)?1:-1);

 if(dx>dy)
{
         int two_dy=(2*dy);
         int two_dy_dx=(2*(dy-dx));
         int p=((2*dy)-dx);

         int x=x1;
         int y=y1;
```

```
          putpixel(x,y,color);

          while(x<x2)
         {
           x++;

           if(p<0)
              p+=two_dy;

           else
             {
             y+=inc_dec;
             p+=two_dy_dx;
              }

           putpixel(x,y,color);
         }
 }

  else
 {
          int two_dx=(2*dx);
          int two_dx_dy=(2*(dx-dy));
          int p=((2*dx)-dy);

          int x=x1;
          int y=y1;
```

```
             putpixel(x,y,color);


             while(y!=y2)
            {
              y+=inc_dec;


              if(p<0)
                 p+=two_dx;


              else
                 {
                x++;
                p+=two_dx_dy;
                 }


              putpixel(x,y,color);
            }
      }
    }


 /************************************************************************///-------------
-----------  show_screen( )  -------------------------
///************************************************************************/void
show_screen( )

void  show_screen( )

{

      setfillstyle(1,1);

     bar(210,60,420,78);


      settextstyle(0,0,1);
 setcolor(15);
   outtextxy(5,65,"*-----------------------                         ------------------------*");
 setcolor(11);
   outtextxy(218,65,"3D Shearing along x-axis");


 setcolor(15);
   outtextxy(5,400,"*-------------------------                      -------------------------*");
 setcolor(12);
   outtextxy(229,400,"Press any Key to exit.");
}
```



```
3D Rotation on given object
Vivek Kumar
17115091
29/08/2019
```

33) scan line algorithm for area filling

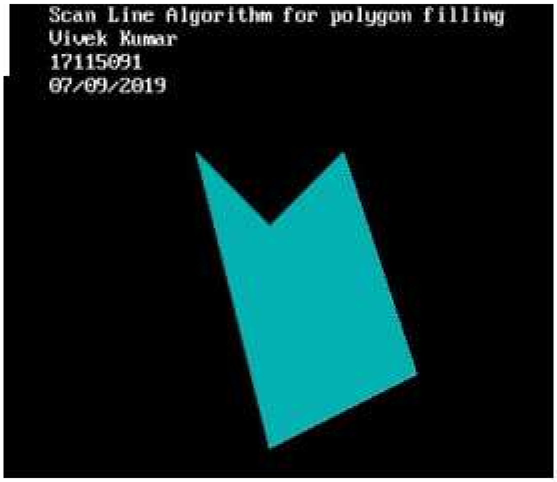
```
#include <graphics.h>

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>
```

```
struct Node
{
   int x;
   int y;
   struct Node* next;
};

void fill (int pt[][2], int clr);
void floodfill4 (int x, int y, int oldclr, int newclr);
void insert (int x, int y, struct Node** last);

void main()
{
   int gd = DETECT, gm;
   int i, j;
   int pt[3][2];
   int clr;
   initgraph (&gd, &gm, "..\\bgi");
   printf ("This program demonstrates filling a polygon.\n");
   printf("Vivek Kumar\n 17115091\n 07/09/2019\n");
   printf ("Enter the x- and y-coordinates for three points:\n");
   for (i=0; i<3; i++)
            for (j=0; j<2; j++)
               scanf ("%d", &pt[i][j]);

   printf ("Enter the fill-colour: (Any number from 1 to 14) ");
   scanf ("%d", &clr);
   fill (pt, clr);
```

```
}

void fill (int pt[][2], int clr)
{
    int seedx, seedy;
    setcolor (WHITE);
    line (pt[0][0], pt[0][1], pt[1][0], pt[1][1]);
    line (pt[1][0], pt[1][1], pt[2][0], pt[2][1]);
    line (pt[2][0], pt[2][1], pt[0][0], pt[0][1]);
    getch();

    seedx = (pt[0][0] + pt[1][0] + pt[2][0]) / 3;
    seedy = (pt[0][1] + pt[1][1] + pt[2][1]) / 3;

    floodfill4 (seedx, seedy, BLACK, clr);
    getch();

    closegraph();
    return;
}

void floodfill4 (int x, int y, int oldclr, int newclr)
{
    struct Node* first, *last, *tmp;

    first = (struct Node*) malloc (sizeof (struct Node));
    if (first == NULL)
    {
            closegraph();
```

```
          fprintf (stderr, "floodfill4: Out of memory.\n");
          exit (2);
}
if (oldclr == newclr)
{
          free (first);
          return;
}

first->x = x;
first->y = y;
first->next = NULL;
last = first;

while (first != NULL)
{
          putpixel (x, y, newclr);

          if (getpixel (x, y-1) == oldclr)
          {
             putpixel (x, y-1, newclr);
             insert (x, y-1, &last);
          }



          if (getpixel (x, y+1) == oldclr)
          {
             putpixel (x, y+1, newclr);
             insert (x, y+1, &last);
```

```
            }

            if (getpixel (x-1, y) == oldclr)
            {
               putpixel (x-1, y, newclr);
               insert (x-1, y, &last);
            }

            if (getpixel (x+1, y) == oldclr)
            {
               putpixel (x+1, y, newclr);
               insert (x+1, y, &last);
            }

            tmp = first;
            first = first->next;
            x = first->x;
            y = first->y;
            free (tmp);
   }
}

void insert (int x, int y, struct Node** last)
{
   struct Node* p;
   p = (struct Node*) malloc (sizeof (struct Node));
   if (p == NULL)
   {
            closegraph();
            fprintf (stderr, "\n insert: Out of memory.\n");

            exit (2);

   }


   p->x = x;

   p->y = y;

   p->next = NULL;

   (*last)->next = p;

   *last = (*last)->next;

}
```

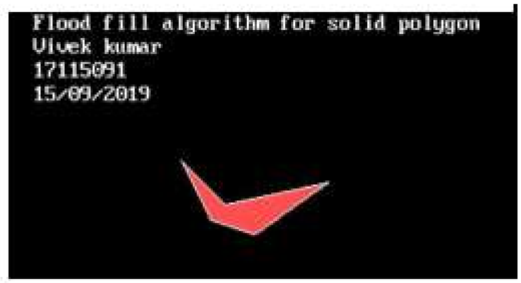


```
Scan Line Algorithm for polygon filling
Vivek Kumar
17115091
07/09/2019
```

34)flood fill algorithm for solid polygon

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<iostream.h>

void flood(int x, int y, int new_col, int old_col)
```

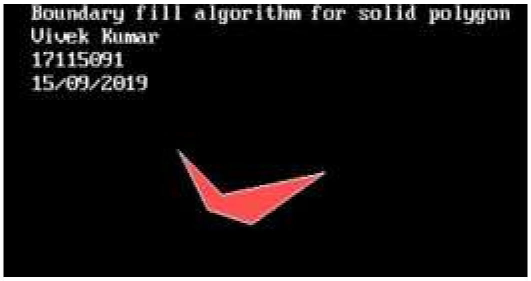
```
{
        if(getpixel(x, y) == old_col)
        {
                putpixel(x, y, new_col);
                flood(x + 1, y, new_col, old_col);
                flood(x - 1, y, new_col, old_col);
                flood(x, y + 1, new_col, old_col);
                flood(x, y - 1, new_col, old_col);
        }
}

int main()
{
        int gd, gm = DETECT;
        initgraph(&gm, &gd, "..\\bgi");
        cout<<"WAP TO IMPLEMENT FLOOD FILL ALGORITM FOR SOLID POLYGON"<<endl;
        cout<<"Vivek Kumar"<<endl;
        cout<<"17115091"<<endl;
        cout<<"07/09/2019"<<endl;
        int top, left, bottom, right;
        top = left = 150;
        bottom = right = 210;
        rectangle(left, top, right, bottom);
        int x = 155;
        int y = 155;
        int newcolor = 12;
        int oldcolor = 0;
        flood(x, y, newcolor, oldcolor);
        getch();
        closegraph();

        return 0;

}
```



```
Flood fill algorithm for solid polygon
Vivek kumar
17115091
15/09/2019
```

35)boundary fill algorithm

```
#include<stdio.h>

#include<graphics.h>

#include<dos.h>

#include<conio.h>

void boundaryfill(int x,int y,int f_color,int b_color)

{

        if(getpixel(x,y)!=b_color && getpixel(x,y)!=f_color)

        {

                putpixel(x,y,f_color);

                boundaryfill(x+1,y,f_color,b_color);

                boundaryfill(x,y+1,f_color,b_color);

                boundaryfill(x-1,y,f_color,b_color);

                boundaryfill(x,y-1,f_color,b_color);

        }

}

int main()

{

        int gm,gd=DETECT,radius;
        int x,y;

        initgraph(&gd,&gm,"..\\bgi");

        printf("WAP TO IMPLEMENT BOUNDARY FILL ALGORITHM\n");

        printf("Vivek Kumar\n 17115091 \n 07/09/2019\n");

        printf("Enter x and y positions for circle\n");

        scanf("%d%d",&x,&y);

        printf("Enter radius of circle\n");

        scanf("%d",&radius);

        circle(x,y,radius);

        boundaryfill(x,y,4,15);

        getch();

        closegraph();


        return 0;

}
```

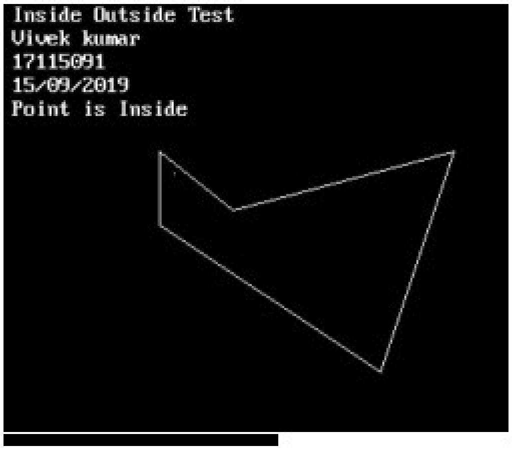


```
Boundary fill algorithm for solid polygon
Vivek Kumar
17115091
15/09/2019
```

36)inside outside test algorithm

```
#include <iostream.h>

#include <graphics.h>

#include <conio.h>

#include <math.h>


int a[]={100,100,150,140,300,100,250,250,100,150,100,100};
int check(int x, int y)
{
    int i,cnt=0,s=sizeof(a)/2;
    for(;x>=0;x--)
    {
        if(getpixel(x,y)==15)
        {
            cnt++;
            for(i=0;i<s;i+=2)
            {
                if(x==a[i] && y==a[i+1])
                {
                    if(!(a[(i+3)%s]>y ^ a[(i-1+s)%s]>y))
                    {
                        cnt++;
                    }
                }

            }
        }
    }
    cout<<cnt<<endl;
    return cnt;
}

void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"..\\bgi");
```

```
        drawpoly(6,a);
        int x=250,y=101,c;
        putpixel(x,y,15);
        c=check(x-1,y);
        if(c%2==0)
        {cout<<"Outside"; }
        else
        {cout<<"Inside"; }
        getch();
        closegraph();
}
```

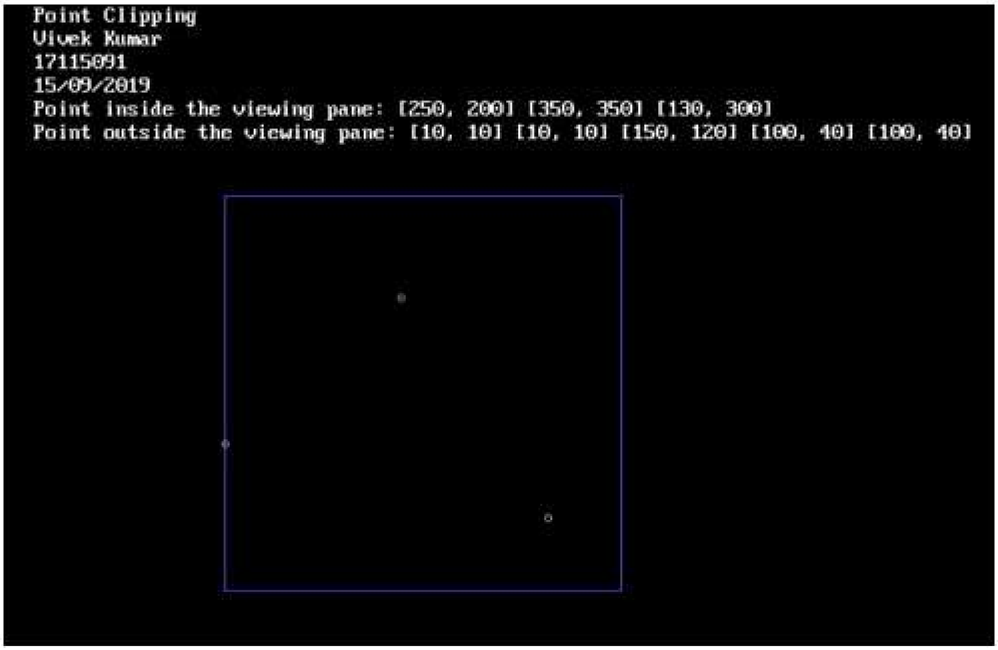


```
Inside Outside Test
Vivek kumar
17115091
15/09/2019
Point is Inside
```

37)point clipping concept

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void pointClip(int XY[][2], int n, int Xmin, int Ymin,int Xmax, int Ymax)
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"..\\bgi");
```

```
printf("WAP TO IMPLEMENT POINT CLIPPING CONCEPT\n");
printf("Vivek Kumar\n 17115091\n 07/09/2019\n");
setcolor(9);
rectangle(130,130,400,400);
setcolor(7);
for (int i=0; i<n; i++)
{
if ( (XY[i][0] >= Xmin) && (XY[i][0] <= Xmax))
{
        if( (XY[i][1] >= Ymin) && (XY[i][1] <= Ymax))
        putpixel(XY[i][0],XY[i][1],9);
}
}
printf ("Point inside the viewing pane:\n");
for (i=0; i<n; i++)
{
        if((XY[i][0] >= Xmin) && (XY[i][0] <= Xmax))
        {
                if((XY[i][1] >= Ymin) && (XY[i][1] <= Ymax)) {
                        printf ("[%d, %d] ", XY[i][0], XY[i][1]);
                        circle(XY[i][0],XY[i][1],2);
                        }
        }
}
printf ("\nPoint outside the viewing pane:\n");
for (i=0; i<n; i++)
{
        if ((XY[i][0] < Xmin) || (XY[i][0] > Xmax))
                printf ("[%d, %d] ", XY[i][0], XY[i][1]);
        if ((XY[i][1] < Ymin) || (XY[i][1] > Ymax))
            printf ("[%d, %d] ", XY[i][0], XY[i][1]);
    }
    getch();
}
int main()
{
    int XY[6][2] = {{10,10}, {250,200}, {350,350},
                    {130,300}, {150,120}, {100,40}};

    int Xmin = 130;
    int Xmax = 400;
    int Ymin = 130;
    int Ymax = 400;
    pointClip(XY, 6, Xmin, Ymin, Xmax, Ymax);
    getch();
    closegraph();
    return 0;
}
```



```
Point Clipping
Vivek Kumar
17115091
15/09/2019
Point inside the viewing pane: [250, 200] [350, 350] [130, 300]
Point outside the viewing pane: [10, 10] [10, 10] [150, 120] [100, 40] [100, 40]
```

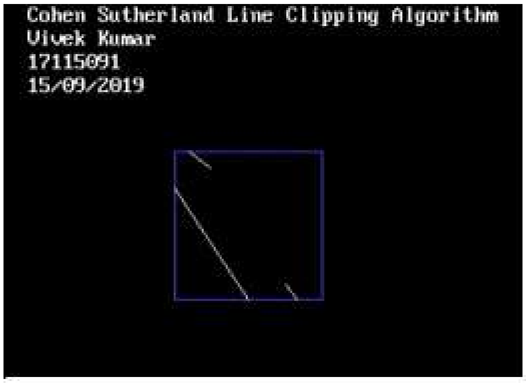
38)cohen Sutherland algorithm

```
#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<graphics.h>


const int INSIDE = 0; // 0000

const int LEFT = 1; // 0001

const int RIGHT = 2; // 0010

const int BOTTOM = 4; // 0100

const int TOP = 8; // 1000

const int x_max = 400;

const int y_max = 400;

const int x_min = 100;

const int y_min = 100;


int computeCode(double x, double y)
```

```
{
    int code = INSIDE;
    if (x < x_min)
    code |= LEFT;
    else if (x > x_max)
        code |= RIGHT;
    if (y < y_min)
        code |= BOTTOM;
    else if (y > y_max)
        code |= TOP;
    return code;
}
void cohenSutherlandClip(double x1, double y1, double x2, double y2)
{
    int code1 = computeCode(x1, y1);
    int code2 = computeCode(x2, y2);
    int accept = 0;
    while (1)
    {
        if ((code1 == 0) && (code2 == 0))
        {
            accept = 1;
            break;
        }
        else if (code1 & code2)
        {
            break;
        }
        else
```

```
{
        int code_out;
        double x, y;
        if (code1 != 0)
                code_out = code1;
        else
                code_out = code2;
        if (code_out & TOP)
        {
                x = x1 + (x2 - x1) * (y_max - y1) / (y2 - y1);
                y = y_max;
        }
        else if (code_out & BOTTOM)
        {
                x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1);
                y = y_min;
        }
        else if (code_out & RIGHT)
        {
                y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1);
                x = x_max;
        }
        else if (code_out & LEFT)
        {
                y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1);
                x = x_min;
        }
        if (code_out == code1)
        {
                x1 = x;
                y1 = y;
                code1 = computeCode(x1, y1);
            }
            else
            {
                x2 = x;
                y2 = y;
                code2 = computeCode(x2, y2);
            }
        }
    }
    if (accept)
    {
            line(x1,y1,x2,y2);
    }
    else
        cout << "Line rejected" << endl;
}

int main()
{   int gd=DETECT,gm;
     initgraph(&gd,&gm,"..\\bgi");
     printf("WAP TO IMPLEMENT COHEN SUTHERLAND ALGO. FOR LINE CLIPPING\n");
     printf("Vivek Kumar\n 17115091\n 07/09/2019\n");
     setcolor(6);
     rectangle(x_min, y_min, x_max, y_max);
     setcolor(7);
    // First Line segment
```

```
        // P11 = (5, 5), P12 = (7, 7)
        cohenSutherlandClip(50, 50, 300, 300);

        // Second Line segment
        // P21 = (7, 9), P22 = (11, 4)
        cohenSutherlandClip(71, 111, 540, 540);

        // Third Line segment
        // P31 = (1, 5), P32 = (4, 1)
        cohenSutherlandClip(10, 98, 284, 351);
        cohenSutherlandClip(100,10,800,910);
        getch();
        closegraph();
        return 0;
}
```



```
Cohen Sutherland Line Clipping Algorithm
Vivek Kumar
17115091
15/09/2019
```

39)mid point algorithm for line clipping

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

```
{
   public:
   float x_1;
   float y_1;
   float x_2;
   float y_2;

   LineCoordinates(const float x1, const float y1,
                              const float x2,const float y2)
        {
       x_1=x1;
       y_1=y1;
       x_2=x2;
       y_2=y2;
        }
  };

 /**********************************************************************///-------------
-----------  WindowCoordinates  ------------------------
///**********************************************************************/
  class WindowCoordinates
  {
    public:
   float x_min;
   float y_min;
   float x_max;
   float y_max;

   WindowCoordinates(const float x1,const float y1,
                          const float x2,const float y2)

         {

        x_min=x1;

        y_min=y1;

        x_max=x2;

        y_max=y2;

         }

  };


/**********************************************************************///-------------
--------------- RegionCode ---------------------------
///**********************************************************************/

class RegionCode

  {

    public:

   int bit_1;

   int bit_2;

   int bit_3;

   int bit_4;


   RegionCode( )

         {

        bit_1=0;

        bit_2=0;

        bit_3=0;

        bit_4=0;

         }


   const int equal_zero( )
```

```
     {
    if(bit_1==0 && bit_2==0 && bit_3==0 && bit_4==0)
      return 1;

    return 0;
     }

void get_logical_AND(RegionCode rc1,RegionCode rc2)
     {
    if(rc1.bit_1==1 && rc2.bit_1==1)
      bit_1=1;

    if(rc1.bit_2==1 && rc2.bit_2==1)
      bit_2=1;

    if(rc1.bit_3==1 && rc2.bit_3==1)
      bit_3=1;

    if(rc1.bit_4==1 && rc2.bit_4==1)
      bit_4=1;
     }

void get_region_code(const WindowCoordinates wc,
                     const int x,const int y)
     {
    if((wc.x_min-x)>0)
      bit_1=1;

    if((x-wc.x_max)>0)
```

```
              bit_2=1;

            if((wc.y_min-y)>0)
              bit_3=1;

            if((y-wc.y_max)>0)
              bit_4=1;
             }
   };


void show_screen( );

const int clip_line(const WindowCoordinates,LineCoordinates&);

void calculate_intersecting_points(const WindowCoordinates,LineCoordinates&);

void Rectangle(const int,const int,const int,const int);
void Line(const int,const int,const int,const int);


int main( )
   {
     int driver=VGA;
     int mode=VGAHI;

     initgraph(&driver,&mode,"..\\Bgi");

     show_screen( );
```

```
  WindowCoordinates WC(180,140,470,340);

  setcolor(15);
Rectangle(WC.x_min,WC.y_min,WC.x_max,WC.y_max);

  LineCoordinates LC_1(150,160,120,320);
  LineCoordinates LC_2(250,150,200,200);
  LineCoordinates LC_3(160,200,490,260);
  LineCoordinates LC_4(300,300,400,380);
  LineCoordinates LC_5(550,300,450,400);
  LineCoordinates LC_6(440,110,400,370);

  setcolor(7);
Line(LC_1.x_1,LC_1.y_1,LC_1.x_2,LC_1.y_2);
Line(LC_2.x_1,LC_2.y_1,LC_2.x_2,LC_2.y_2);
Line(LC_3.x_1,LC_3.y_1,LC_3.x_2,LC_3.y_2);
Line(LC_4.x_1,LC_4.y_1,LC_4.x_2,LC_4.y_2);
Line(LC_5.x_1,LC_5.y_1,LC_5.x_2,LC_5.y_2);
Line(LC_6.x_1,LC_6.y_1,LC_6.x_2,LC_6.y_2);

  char Key=NULL;

  do
 {
      Key=getch( );
 }
  while(Key!='C' && Key!='c');
```

```
   settextstyle(0,0,1);
 setcolor(0);
   outtextxy(163,450,"  Press 'C' to see the Clipped Lines.  ");

setcolor(15);
   outtextxy(163,450,"------                         -------");

 setcolor(12);
   outtextxy(213,450,"  Press any Key to exit.  ");

   setcolor(10);

   if(clip_line(WC,LC_1))
  Line(LC_1.x_1,LC_1.y_1,LC_1.x_2,LC_1.y_2);

   if(clip_line(WC,LC_2))
  Line(LC_2.x_1,LC_2.y_1,LC_2.x_2,LC_2.y_2);

   if(clip_line(WC,LC_3))
  Line(LC_3.x_1,LC_3.y_1,LC_3.x_2,LC_3.y_2);

   if(clip_line(WC,LC_4))
  Line(LC_4.x_1,LC_4.y_1,LC_4.x_2,LC_4.y_2);

   if(clip_line(WC,LC_5))
  Line(LC_5.x_1,LC_5.y_1,LC_5.x_2,LC_5.y_2);

   if(clip_line(WC,LC_6))
  Line(LC_6.x_1,LC_6.y_1,LC_6.x_2,LC_6.y_2);
```

```
      getch( );

      return 0;

   }


 /*********************************************************************///---------------
-------------  clip_line( )  ----------------------------
///*********************************************************************/

 const int clip_line(const WindowCoordinates wc,LineCoordinates &lc)

   {

      RegionCode rc1,rc2,rc;

      rc1.get_region_code(wc,lc.x_1,lc.y_1);

      rc2.get_region_code(wc,lc.x_2,lc.y_2);

      rc.get_logical_AND(rc1,rc2);


      if(rc1.equal_zero( ) && rc2.equal_zero( ))

     return 1;

      else if(!rc.equal_zero( ))

     return 0;

      else

     {

            calculate_intersecting_points(wc,lc);


            lc.x_1=(int)(lc.x_1+0.5);

            lc.y_1=(int)(lc.y_1+0.5);

            lc.x_2=(int)(lc.x_2+0.5);

            lc.y_2=(int)(lc.y_2+0.5);


            if(lc.x_1==lc.x_2 && lc.y_1==lc.y_2)
            return 0;
        }


       return 1;
     }


 /**********************************************************************///-------------
---  calculate_intersecting_points( )  ------------------
///**********************************************************************/

 void calculate_intersecting_points(const WindowCoordinates wc,
                                            LineCoordinates &lc)
     {
       LineCoordinates lc1(lc.x_1,lc.y_1,lc.x_2,lc.y_2);
       LineCoordinates lc2(lc.x_2,lc.y_2,lc.x_1,lc.y_1);

       float x_mid;
       float y_mid;

       if(lc1.y_1>wc.y_max)
      {
            while(lc1.y_1!=wc.y_max)
            {
              x_mid=((lc1.x_1+lc1.x_2)/2);
              y_mid=((lc1.y_1+lc1.y_2)/2);

              if(y_mid>=wc.y_max)
                {
               lc1.x_1=x_mid;
               lc1.y_1=y_mid;
              }

           else
              {
             lc1.x_2=x_mid;
             lc1.y_2=y_mid;
              }

           if((int)(lc1.x_1+0.5)==(int)(lc1.x_2+0.5) &&
                           (int)(lc1.y_1+0.5)==(int)(lc1.y_2+0.5))
              break;
        }
}

 else if(lc1.y_1<wc.y_min)
{
         while(lc1.y_1!=wc.y_min)
        {
           x_mid=((lc1.x_1+lc1.x_2)/2);
           y_mid=((lc1.y_1+lc1.y_2)/2);

           if(y_mid<=wc.y_min)
              {
             lc1.x_1=x_mid;
             lc1.y_1=y_mid;
              }

           else
              {
```

```
        lc1.x_2=x_mid;
        lc1.y_2=y_mid;
         }

       if((int)(lc1.x_1+0.5)==(int)(lc1.x_2+0.5) &&
                (int)(lc1.y_1+0.5)==(int)(lc1.y_2+0.5))
         break;
     }
}

 if(lc1.x_1>wc.x_max)
{
      while(lc1.x_1!=wc.x_max)
     {
       x_mid=((lc1.x_1+lc1.x_2)/2);
       y_mid=((lc1.y_1+lc1.y_2)/2);

       if(x_mid>=wc.x_max)
         {
        lc1.x_1=x_mid;
        lc1.y_1=y_mid;
         }

       else
         {
        lc1.x_2=x_mid;
        lc1.y_2=y_mid;
         }
      if((int)(lc1.x_1+0.5)==(int)(lc1.x_2+0.5) &&
                (int)(lc1.y_1+0.5)==(int)(lc1.y_2+0.5))
        break;
     }
}

 else if(lc1.x_1<wc.x_min)
{
     while(lc1.x_1!=wc.x_min)
     {
      x_mid=((lc1.x_1+lc1.x_2)/2);
      y_mid=((lc1.y_1+lc1.y_2)/2);

      if(x_mid<=wc.x_min)
        {
       lc1.x_1=x_mid;
       lc1.y_1=y_mid;
        }

      else
        {
       lc1.x_2=x_mid;
       lc1.y_2=y_mid;
        }

      if((int)(lc1.x_1+0.5)==(int)(lc1.x_2+0.5) &&
                (int)(lc1.y_1+0.5)==(int)(lc1.y_2+0.5))
        break;
     }
```

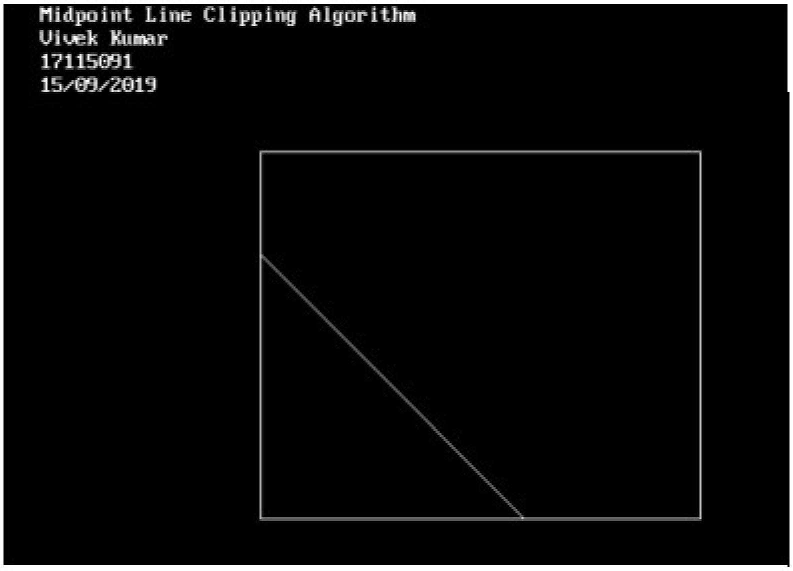
```
}

 lc2.x_2=lc1.x_1;
 lc2.y_2=lc1.y_1;

 if(lc2.y_1>wc.y_max)
{
     while(lc2.y_1!=wc.y_max)
    {
      x_mid=((lc2.x_1+lc2.x_2)/2);
      y_mid=((lc2.y_1+lc2.y_2)/2);

      if(y_mid>=wc.y_max)
        {
       lc2.x_1=x_mid;
       lc2.y_1=y_mid;
        }

      else
        {
       lc2.x_2=x_mid;
       lc2.y_2=y_mid;
        }

      if((int)(lc2.x_1+0.5)==(int)(lc2.x_2+0.5) &&
                (int)(lc2.y_1+0.5)==(int)(lc2.y_2+0.5))
        break;
    }
}
```

```
 else if(lc2.y_1<wc.y_min)
{
         while(lc2.y_1!=wc.y_min)
        {
          x_mid=((lc2.x_1+lc2.x_2)/2);
          y_mid=((lc2.y_1+lc2.y_2)/2);

          if(y_mid<=wc.y_min)
             {
            lc2.x_1=x_mid;
            lc2.y_1=y_mid;
             }

          else
             {
            lc2.x_2=x_mid;
            lc2.y_2=y_mid;
             }

          if((int)(lc2.x_1+0.5)==(int)(lc2.x_2+0.5) &&
                          (int)(lc2.y_1+0.5)==(int)(lc2.y_2+0.5))
             break;
        }
}

 if(lc2.x_1>wc.x_max)
{
         while(lc2.x_1!=wc.x_max)
```

```
      {
        x_mid=((lc2.x_1+lc2.x_2)/2);
        y_mid=((lc2.y_1+lc2.y_2)/2);

        if(x_mid>=wc.x_max)
          {
         lc2.x_1=x_mid;
         lc2.y_1=y_mid;
          }

        else
          {
         lc2.x_2=x_mid;
         lc2.y_2=y_mid;
          }

        if((int)(lc2.x_1+0.5)==(int)(lc2.x_2+0.5) &&
                   (int)(lc2.y_1+0.5)==(int)(lc2.y_2+0.5))
          break;
      }
}

 else if(lc2.x_1<wc.x_min)
{
       while(lc2.x_1!=wc.x_min)
      {
        x_mid=((lc2.x_1+lc2.x_2)/2);
        y_mid=((lc2.y_1+lc2.y_2)/2);
```

```
            if(x_mid<=wc.x_min)
              {
             lc2.x_1=x_mid;
             lc2.y_1=y_mid;
              }

            else
              {
             lc2.x_2=x_mid;
             lc2.y_2=y_mid;
              }

            if((int)(lc2.x_1+0.5)==(int)(lc2.x_2+0.5) &&
                       (int)(lc2.y_1+0.5)==(int)(lc2.y_2+0.5))
              break;
          }
    }

     lc.x_1=lc1.x_1;
     lc.y_1=lc1.y_1;
     lc.x_2=lc2.x_1;
     lc.y_2=lc2.y_1;
   }

/************************************************************************///--------------
-------------  Rectangle( )  ---------------------------
///************************************************************************/

void Rectangle(const int x_1,const int y_1,const int x_2,const int y_2)
   {
     Line(x_1,y_1,x_2,y_1);

     Line(x_2,y_1,x_2,y_2);

     Line(x_2,y_2,x_1,y_2);

     Line(x_1,y_2,x_1,y_1);

   }


 /************************************************************************///--------------
-----------------  Line( )  ----------------------------
///************************************************************************/

   void Line(const int x_1,const int y_1,const int x_2,const int y_2)

   {

     int color=getcolor( );


     int x1=x_1;

     int y1=y_1;


     int x2=x_2;

     int y2=y_2;


     if(x_1>x_2)

    {

           x1=x_2;

           y1=y_2;


           x2=x_1;

           y2=y_1;

    }


     int dx=abs(x2-x1);
     int dy=abs(y2-y1);
     int inc_dec=((y2>=y1)?1:-1);

     if(dx>dy)
    {
           int two_dy=(2*dy);
           int two_dy_dx=(2*(dy-dx));
           int p=((2*dy)-dx);

           int x=x1;
           int y=y1;

           putpixel(x,y,color);

           while(x<x2)
          {
             x++;

             if(p<0)
                p+=two_dy;

             else
                {
               y+=inc_dec;
               p+=two_dy_dx;
                }

             putpixel(x,y,color);
          }
    }

     else
    {
           int two_dx=(2*dx);
           int two_dx_dy=(2*(dx-dy));
           int p=((2*dx)-dy);

           int x=x1;
           int y=y1;

           putpixel(x,y,color);

           while(y!=y2)
          {
            y+=inc_dec;

            if(p<0)
               p+=two_dx;

            else
              {
             x++;
             p+=two_dx_dy;
              }

            putpixel(x,y,color);
          }
    }
```

```
   }


 /*********************************************************************///--------------
------------  show_screen( )  --------------------------
///*********************************************************************/

 void show_screen( )

 {

    setfillstyle(1,1);

    bar(60,26,565,38);

    settextstyle(0,0,1);

    setcolor(11);

    outtextxy(68,29,"Cohen-Sutherland MidPoint SubDivision Line Clipping Algorithm");

    outtextxy(68,45,"WAP TO IMPLEMENT MID POINT ALGORITHM FOR LINE CLIPPING");

    outtextxy(68,58,"AMIT KUMAR, 17115010, 05/09/2019");

    setcolor(5);

    outtextxy(163,450,"  Press 'C' to see the Clipped Lines.  ");

    setcolor(7);

 }
```



```
Midpoint Line Clipping Algorithm
Vivek Kumar
17115091
15/09/2019
```

40)Liang Barsky algorithm for line clipping

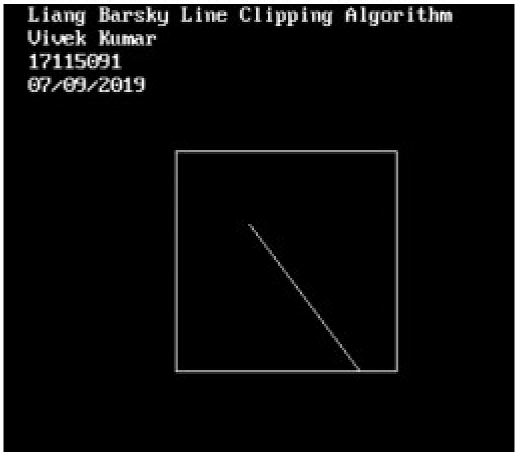
```
#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>


class LineCoordinates

{

     public:

     float x_1;

     float y_1;

     float x_2;

     float y_2;

     LineCoordinates(const float x1,const float y1, const float x2,const float y2)

             {

            x_1=x1;

       y_1=y1;

       x_2=x2;

       y_2=y2;

        }

   };


 /**********************************************************************///--------------
-----------  WindowCoordinates  -------------------------
///**********************************************************************/

class WindowCoordinates

   {

      public:
```

```
   float x_min;
   float y_min;
   float x_max;
   float y_max;

   WindowCoordinates(const float x1,const float y1,
                          const float x2,const float y2)
        {
        x_min=x1;
    y_min=y1;
    x_max=x2;
    y_max=y2;
     }
  };


void show_screen( );

const int clip_line(const WindowCoordinates,LineCoordinates&);
const int check_line(const float,const float,float&,float&);

void Rectangle(const int,const int,const int,const int);
void Line(const int,const int,const int,const int);


int main( )
  {
    int driver=VGA;
    int mode=VGAHI;
  initgraph(&driver,&mode,"..\\Bgi");

  show_screen( );

  WindowCoordinates WC(180,140,470,340);

  setcolor(15);
Rectangle(WC.x_min,WC.y_min,WC.x_max,WC.y_max);

  LineCoordinates LC_1(150,160,120,320);
  LineCoordinates LC_2(250,150,200,200);
  LineCoordinates LC_3(160,200,490,260);
  LineCoordinates LC_4(300,300,400,380);
  LineCoordinates LC_5(550,300,450,400);
  LineCoordinates LC_6(440,110,400,370);

  setcolor(7);
Line(LC_1.x_1,LC_1.y_1,LC_1.x_2,LC_1.y_2);
Line(LC_2.x_1,LC_2.y_1,LC_2.x_2,LC_2.y_2);
Line(LC_3.x_1,LC_3.y_1,LC_3.x_2,LC_3.y_2);
Line(LC_4.x_1,LC_4.y_1,LC_4.x_2,LC_4.y_2);
Line(LC_5.x_1,LC_5.y_1,LC_5.x_2,LC_5.y_2);
Line(LC_6.x_1,LC_6.y_1,LC_6.x_2,LC_6.y_2);

  char Key=NULL;

  do
 {
     Key=getch( );
  }
   while(Key!='C' && Key!='c');

   settextstyle(0,0,1);
 setcolor(0);
   outtextxy(163,450,"  Press 'C' to see the Clipped Lines.  ");

setcolor(15);
   outtextxy(163,450,"------                        -------");

 setcolor(12);
   outtextxy(213,450,"  Press any Key to exit.  ");

   setcolor(10);

   if(clip_line(WC,LC_1))
  Line(LC_1.x_1,LC_1.y_1,LC_1.x_2,LC_1.y_2);

   if(clip_line(WC,LC_2))
  Line(LC_2.x_1,LC_2.y_1,LC_2.x_2,LC_2.y_2);

   if(clip_line(WC,LC_3))
  Line(LC_3.x_1,LC_3.y_1,LC_3.x_2,LC_3.y_2);

   if(clip_line(WC,LC_4))
  Line(LC_4.x_1,LC_4.y_1,LC_4.x_2,LC_4.y_2);

   if(clip_line(WC,LC_5))
```

```
     Line(LC_5.x_1,LC_5.y_1,LC_5.x_2,LC_5.y_2);


     if(clip_line(WC,LC_6))
     Line(LC_6.x_1,LC_6.y_1,LC_6.x_2,LC_6.y_2);


     getch( );
     return 0;
   }


 /*************************************************************************///--------------
-------------  clip_line( )  ----------------------------
///*************************************************************************/
 const int clip_line(const WindowCoordinates wc,LineCoordinates &lc)
   {
     float u_1=0;
     float u_2=1;


     float dx=(lc.x_2-lc.x_1);
     float dy=(lc.y_2-lc.y_1);


     float p1=(-dx);
     float p2=dx;
     float p3=(-dy);
     float p4=dy;


     float q1=(lc.x_1-wc.x_min);
     float q2=(wc.x_max-lc.x_1);
     float q3=(lc.y_1-wc.y_min);
     float q4=(wc.y_max-lc.y_1);
   if(check_line(p1,q1,u_1,u_2) && check_line(p2,q2,u_1,u_2) &&
        check_line(p3,q3,u_1,u_2) && check_line(p4,q4,u_1,u_2))
  {
     if(u_2<1)
    {
       lc.x_2=(lc.x_1+(u_2*dx));
       lc.y_2=(lc.y_1+(u_2*dy));
    }

     if(u_1>0)
    {
       lc.x_1+=(u_1*dx);
       lc.y_1+=(u_1*dy);
    }

     lc.x_1=(int)(lc.x_1+0.5);
     lc.y_1=(int)(lc.y_1+0.5);
     lc.x_2=(int)(lc.x_2+0.5);
     lc.y_2=(int)(lc.y_2+0.5);

     return 1;
  }

   return 0;
}
```

```
/**********************************************************************///--------------
----------------  check_line( )  ------------------------
///**********************************************************************/

const int check_line(const float p,const float q,float &u_1,float &u_2)

  {

     int flag=1;


     float r=(q/p);


     if(p<0)

    {

      if(r>u_2)

     flag=0;


            else if(r>u_1)

     u_1=r;

    }


     else if(p>0)

    {

      if(r<u_1)

     flag=0;


            else if(r<u_2)

     u_2=r;

    }


     else

    {
      if(q<0)
     flag=0;
    }

     return flag;
  }

/**********************************************************************///--------------
-------------  Rectangle( )  ----------------------------
///**********************************************************************/
void Rectangle(const int x_1,const int y_1,const int x_2,const int y_2)
  {
     Line(x_1,y_1,x_2,y_1);
     Line(x_2,y_1,x_2,y_2);
     Line(x_2,y_2,x_1,y_2);
     Line(x_1,y_2,x_1,y_1);
  }

/**********************************************************************///--------------
-----------  Line( )  -----------------------
///**********************************************************************/
void Line(const int x_1,const int y_1,const int x_2,const int y_2)
  {
     int color=getcolor( );

     int x1=x_1;
     int y1=y_1;

     int x2=x_2;
     int y2=y_2;
 if(x_1>x_2)
{
   x1=x_2;
   y1=y_2;

   x2=x_1;
   y2=y_1;
}

 int dx=abs(x2-x1);
 int dy=abs(y2-y1);
 int inc_dec=((y2>=y1)?1:-1);

 if(dx>dy)
{
   int two_dy=(2*dy);
   int two_dy_dx=(2*(dy-dx));
   int p=((2*dy)-dx);

   int x=x1;
   int y=y1;

   putpixel(x,y,color);

   while(x<x2)
  {
     x++;
     if(p<0)
        p+=two_dy;

     else
        {
       y+=inc_dec;
       p+=two_dy_dx;
        }

     putpixel(x,y,color);
  }
}

 else
{
   int two_dx=(2*dx);
   int two_dx_dy=(2*(dx-dy));
   int p=((2*dx)-dy);

   int x=x1;
   int y=y1;

   putpixel(x,y,color);

   while(y!=y2)
  {
     y+=inc_dec;

     if(p<0)
```

```
          p+=two_dx;

        else
          {
         x++;
         p+=two_dx_dy;
          }

        putpixel(x,y,color);
      }
    }
  }

 /**********************************************************************///--------------
------------  show_screen( )  --------------------------
///**********************************************************************/

 void show_screen( )
   {
     setfillstyle(1,1);
    bar(165,26,470,38);
    settextstyle(0,0,1);
    setcolor(11);
    outtextxy(174,29,"Liang-Barsky Line Clipping Algorithm");
    outtextxy(174,45,"AMIT KUMAR, 17115010, 05/09/2019");
    setcolor(12);
      outtextxy(163,450,"  Press 'C' to see the Clipped Lines.  ");
   }
```



```
Liang Barsky Line Clipping Algorithm
Vivek Kumar
17115091
07/09/2019
```

41)Sutherland polygon clipping algorithm

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include<graphics.h>
#define ROUND(a) ((int)(a+0.5))
#define n 4

#define LEFT_EDGE 0x1
#define RIGHT_EDGE 0x2
#define BOTTOM_EDGE 0x4
#define TOP_EDGE 0x8

#define INSIDE(a)  (!a)
#define REJECT(a,b) (a&b)
#define ACCEPT(a,b) (!(a|b))

typedef struct wcpt2
```

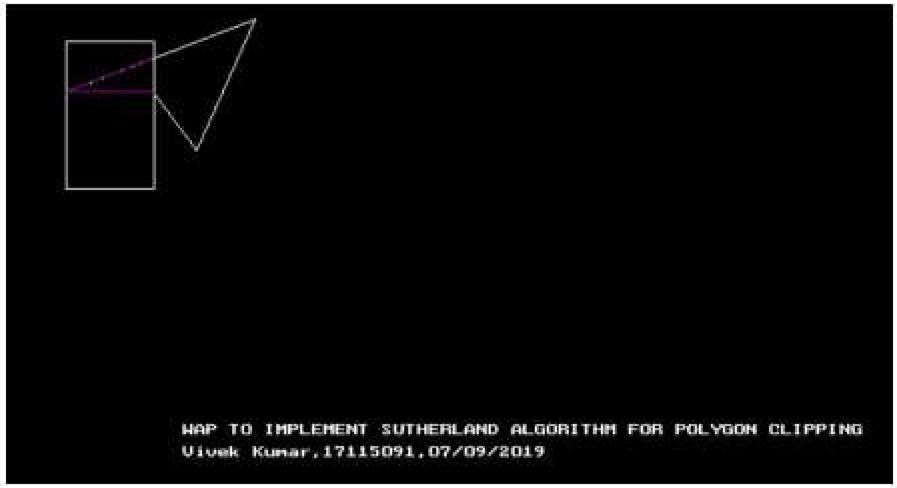
```
{
   int x,y;
}wcpt2;

typedef struct dcpt
{
   int x,y;
}dcpt;

void main()
{
   int gd=DETECT,gm;
   int left,top,right,bottom;
   int x1,x2,y1,y2;
   int maxx, maxy;
     /* our polygon array */int poly[10];
   void clipline(dcpt,dcpt,wcpt2,wcpt2);
   clrscr();

   initgraph(&gd,&gm,"..\\bgi");
   outtextxy(100,285,"WAP TO IMPLEMENT SUTHERLAND ALGORITHM FOR POLYGON CLIPPING");
   outtextxy(100,300,"AMIT KUMAR, 17115010, 05/09/2019");
   maxx = getmaxx()/4;
   maxy = getmaxy()/4;

   poly[0] = 20;       /* 1st vertex */
   poly[1] = maxy / 2;

   poly[2] = maxx - 10; /* 2nd */
   poly[3] = 10;

   poly[4] = maxx - 50; /* 3rd */
   poly[5] = maxy - 20;

   poly[6] = maxx / 2;  /* 4th */
   poly[7] = maxy / 2;

/*   drawpoly doesn't automatically close   the polygon, so we close it.*/
   poly[8] = poly[0];
   poly[9] = poly[1];

  /* draw the polygon */
   drawpoly(5, poly);

   rectangle(20,25,80,125);
   wcpt2 pt1,pt2;
   dcpt winmin,winmax;

   winmin.x=20;
   winmin.y=25;
   winmax.x=80;
   winmax.y=125;

   pt1.x=20;
   pt1.y=maxy/2;
   pt2.x=maxx-10;
   pt2.y=10;
```

```
//    clipline(winmin,winmax,pt1,pt2);
    int i=0;
    for(int index=0;index<n;index++)
    {
                              if(index==n-1)
                              {
                                  pt1.x=poly[i];
                                  pt1.y=poly[i+1];
                                  i=0;
                                  pt2.x=poly[i];
                                  pt2.y=poly[i+1];
                                  clipline(winmin,winmax,pt1,pt2);
                              }
                              else
                              {
                                  pt1.x=poly[i];
                                  pt1.y=poly[i+1];
                                  pt2.x=poly[i+2];
                                  pt2.y=poly[i+3];
                                  clipline(winmin,winmax,pt1,pt2);
                              }
                              i+=2;
    }
    pt1.x=poly[i];
    pt1.y=poly[i+1];
    clipline(winmin,winmax,pt1,pt2);
    getch();
}
```

```
unsigned char encode(wcpt2 pt,dcpt winmin,dcpt winmax)
{
   unsigned char code=0x00;
   if(pt.x < winmin.x)
                   code=code | LEFT_EDGE;
   if(pt.x > winmax.x)
                   code=code | RIGHT_EDGE;
   if(pt.y < winmin.y)
                   code=code | TOP_EDGE;
   if(pt.y > winmax.y)
                   code=code | BOTTOM_EDGE;
   return code;
}


void swappts(wcpt2 *p1,wcpt2 *p2)
{
   wcpt2 tmp;
   tmp = *p1;
   *p1 = *p2;
   *p2 = tmp;
}


void swapcode(unsigned char *c1,unsigned char *c2)
{
   unsigned char tmp;
   tmp = *c1;
   *c1 = *c2;
   *c2 = tmp;
}


void clipline(dcpt winmin,dcpt winmax,wcpt2 p1,wcpt2 p2)
{
   unsigned char encode(wcpt2,dcpt,dcpt);
   unsigned char code1,code2;
   int done = 0 , draw = 0;
   float m;
   void swapcode(unsigned char *c1,unsigned char *c2);
   void swappts(wcpt2 *p1,wcpt2 *p2);

   while(!done)
   {
      code1 = encode(p1,winmin,winmax);
      code2 = encode(p2,winmin,winmax);
      if(ACCEPT(code1,code2))
      {
         draw = 1;
         done = 1;
      }
                        else if(REJECT(code1,code2))
                           done = 1;
                        else if(INSIDE(code1))
      {
            swappts(&p1,&p2);
            swapcode(&code1,&code2);
```

```
  }
  if(code1 & LEFT_EDGE)
  {
      p1.y += (winmin.x - p1.x) *  (p2.y - p1.y) / (p2.x - p1.x);
      p1.x = winmin.x;
  }
              else if(code1 & RIGHT_EDGE)
              {
                    p1.y += (winmax.x - p1.x) *  (p2.y - p1.y) / (p2.x - p1.x);
                    p1.x = winmax.x;
              }
              else if(code1 & TOP_EDGE)
  {
    if(p2.x != p1.x)
      p1.x += (winmin.y - p1.y) *  (p2.x - p1.x) / (p2.y - p1.y);
      p1.y = winmin.y;
  }
              else if(code1 & BOTTOM_EDGE)
  {
    if(p2.x != p1.x)
      p1.x += (winmax.y - p1.y) *  (p2.x - p1.x) / (p2.y - p1.y);
      p1.y = winmax.y;
  }
}
if(draw)
{
setcolor(5);
line(p1.x,p1.y,p2.x,p2.y);
}
```

```
}
```



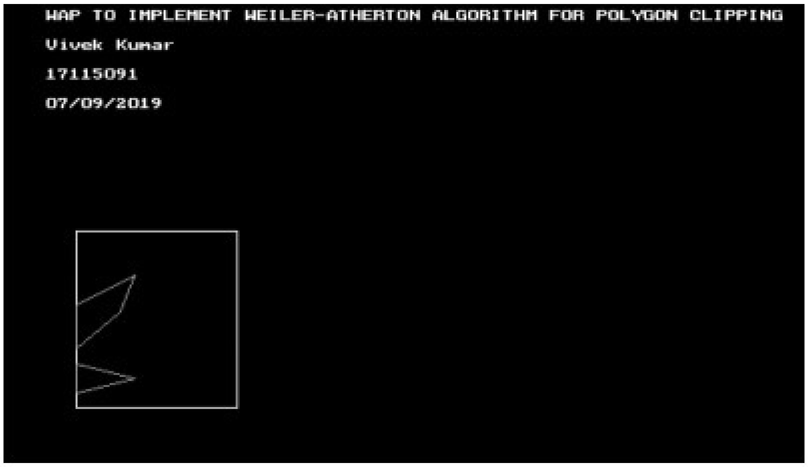
WAP TO IMPLEMENT SUTHERLAND ALGORITHM FOR POLYGON CLIPPING
Vivek Kumar,17115091,07/09/2019

42)weiler Atherton algorithm

```
#include<conio.h>
#include <graphics.h>
#include<dos.h>
void weiler_polygon_clipping();
void main()
{ int gd = DETECT, gm;
  initgraph(&gd,&gm,"..\\BGI");
  clrscr();
  outtextxy(50,80,"Weiler-Atherton Polygon Clipping");
  cleardevice();
  setbkcolor(9);
  weiler_polygon_clipping();
  getch();
  closegraph();
}
```

```
void weiler_polygon_clipping()
{
 outtextxy(50,90,"WAP TO IMPLEMENT WEILER-ATHERTON ALGORITHM FOR POLYGON CLIPPING");
 outtextxy(50,110,"AMIT KUMAR");
 outtextxy(50,130,"17115010");
 outtextxy(50,150,"05/09/2019");
 rectangle(70,240,180,360);
 delay(11);
 line(30,310,110,270);
 delay(1100);
 line(110,270,100,295);
 delay(1100);
 line(100,295,50,330);
 delay(1100);
 line(50,330,110,340);
 delay(1100);
 line(110,340,30,350);
 delay(1100);
 line(30,310,30,350);
 delay(1100);
 outtextxy(20,310,"v1");
 delay(1100);
 outtextxy(110,270,"v2");
 delay(1100);
 outtextxy(105,295,"v3");
 delay(1100);
 outtextxy(45,330,"v4");
 delay(1100);
 outtextxy(115,340,"v5");
```

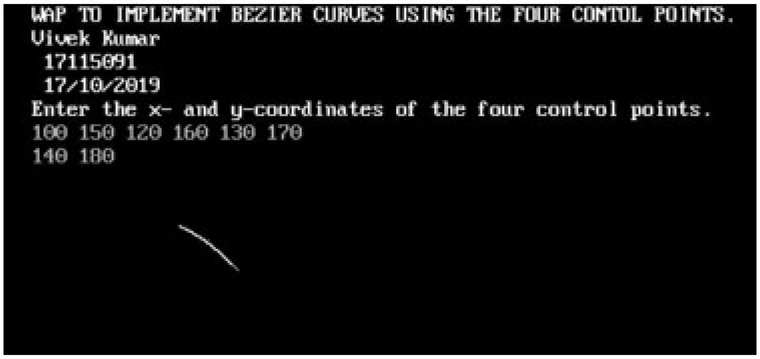
```
  delay(1100);
  outtextxy(20,350,"v6");
  delay(1100);
  outtextxy(65,285,"v1'");
  delay(1100);
  outtextxy(65,305,"v3'");
  delay(1100);
  outtextxy(75,325,"v4'");
  delay(1100);
  outtextxy(50,350,"v5'");
  outtextxy(50,409,"Hit any key to continue...");
  getch();
  cleardevice();
  rectangle(70,240,180,360);
  setcolor(7);
  line(70,290,110,270);
  line(110,270,100,295);
  line(100,295,70,320);
  line(70,290,70,320);
  delay(2000);
  line(70,330,110,340);
  line(70,330,110,340);
  line(110,340,70,350);
  line(70,330,70,350);
  setcolor(13);
  outtextxy(50,409,"Hit any key to continue...");
  getch();
}
```



```
WAP TO IMPLEMENT WEILER-ATHERTON ALGORITHM FOR POLYGON CLIPPING
Vivek Kumar
17115091
07/09/2019
```

43)beizer curve using control points

```
#include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

#include <math.h>

#include<conio.h>

void bezier (int x[4], int y[4])

{

   int i;

   double t;

   for (t = 0.0; t < 1.0; t += 0.0005)

   {

                    double xt = pow (1-t, 3) * x[0] + 3 * t * pow (1-t, 2) * x[1] +

                              3 * pow (t, 2) * (1-t) * x[2] + pow (t, 3) * x[3];


                    double yt = pow (1-t, 3) * y[0] + 3 * t * pow (1-t, 2) * y[1] +
                                        3 * pow (t, 2) * (1-t) * y[2] + pow (t, 3) * y[3];


                        putpixel (xt, yt, WHITE);
    }


    for (i=0; i<4; i++)
                        putpixel (x[i], y[i], YELLOW);


    getch();
    closegraph();
    return;
}


void main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"..\\bgi");
    int x[4], y[4];
    int i;
    printf("WAP TO IMPLEMENT BEZIER CURVES USING THE FOUR CONTOL POINTS.\n");
    printf("Vivek Kumar\n 17115091\n 17/10/2019\n");
    printf ("Enter the x- and y-coordinates of the four control points.\n");
    for (i=0; i<4; i++)
                        scanf ("%d%d", &x[i], &y[i]);
    bezier (x, y);
}
```



```
WAP TO IMPLEMENT BEZIER CURVES USING THE FOUR CONTOL POINTS.
Vivek Kumar
 17115091
 17/10/2019
Enter the x- and y-coordinates of the four control points.
100 150 120 160 130 170
140 180
```

44)

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<math.h>



 /* manipulates the position of planets on the orbit */

 void planetMotion(int xrad, int yrad, int midx, int midy, int x[60], int y[60]) {

                        int i, j = 0;


                        /* positions of planets in their corresponding orbits */
```

```
                for (i = 360; i > 0; i = i - 6) {
                        x[j] = midx - (xrad * cos((i * 3.14) / 180));
                        y[j++] = midy - (yrad * sin((i * 3.14) / 180));
                }
return;
}

int main() {
                /* request auto detection */
                int gdriver = DETECT, gmode, err;
                int i = 0, midx, midy;
                int xrad[9], yrad[9], x[9][60], y[9][60];
                int pos[9], planet[9], tmp;

                /* initialize graphic mode */
                initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");
                err = graphresult();

                if (err != grOk) {
                        /* error occurred */
                        printf("Graphics Error: %s",
                                        grapherrormsg(err));
                        return 0;
                }

                /* mid positions at x and y-axis */
                midx = getmaxx() / 2;
                midy = getmaxy() / 2;
```

```
/* manipulating radius of all 9 planets */
planet[0] = 7;
for (i = 1; i < 9; i++) {
        planet[i] = planet[i - 1] + 1;
}

/* offset position for the planets on their corresponding orbit */
for (i = 0; i < 9; i++) {
        pos[i] = i * 6;
}

/* orbits for all 9 planets */
xrad[0] = 60, yrad[0] = 30;
for (i = 1; i < 9; i++) {
        xrad[i] = xrad[i - 1] + 30;
        yrad[i] = yrad[i - 1] + 15;
}

/* positions of planets on their corresponding orbits */
for (i = 0; i < 9; i++) {
        planetMotion(xrad[i], yrad[i], midx, midy, x[i], y[i]);
}

while (!kbhit()) {
        /* drawing 9 orbits */
        setcolor(WHITE);
        for (i = 0; i < 9; i++) {
                ellipse(midx, midy, 0, 360, xrad[i], yrad[i]);
        }
```

```
/* sun at the mid of the solar system */

setcolor(YELLOW);

setfillstyle(SOLID_FILL, YELLOW);

circle(midx, midy, 20);

floodfill(midx, midy, YELLOW);


/* mercury in first orbit */

setcolor(CYAN);

setfillstyle(SOLID_FILL, CYAN);

pieslice(x[0][pos[0]], y[0][pos[0]], 0, 360, planet[0]);



/* venus in second orbit */

setcolor(GREEN);

setfillstyle(SOLID_FILL, GREEN);

pieslice(x[1][pos[1]], y[1][pos[1]], 0, 360, planet[1]);


/* earth in third orbit */

setcolor(BLUE);

setfillstyle(SOLID_FILL, BLUE);

pieslice(x[2][pos[2]], y[2][pos[2]], 0, 360, planet[2]);


/* mars in fourth orbit */

setcolor(RED);

setfillstyle(SOLID_FILL, RED);

pieslice(x[3][pos[3]], y[3][pos[3]], 0, 360, planet[3]);


/* jupiter in fifth orbit */
```

```
setcolor(BROWN);
setfillstyle(SOLID_FILL, BROWN);
pieslice(x[4][pos[4]], y[4][pos[4]], 0, 360, planet[4]);

/* saturn in sixth orbit */
setcolor(LIGHTGRAY);
setfillstyle(SOLID_FILL, LIGHTGRAY);
pieslice(x[5][pos[5]], y[5][pos[5]], 0, 360, planet[5]);

/* uranus in sevth orbit */
setcolor(BROWN);
setfillstyle(SOLID_FILL, BROWN);
pieslice(x[6][pos[6]], y[6][pos[6]], 0, 360, planet[6]);

/* neptune in eigth orbit */
setcolor(LIGHTBLUE);
setfillstyle(SOLID_FILL, LIGHTBLUE);
pieslice(x[7][pos[7]], y[7][pos[7]], 0, 360, planet[7]);

/* pluto in ninth orbit */
setcolor(LIGHTRED);
setfillstyle(SOLID_FILL, LIGHTRED);
pieslice(x[8][pos[8]], y[8][pos[8]], 0, 360, planet[8]);

/* checking for one complete rotation */
for (i = 0; i < 9; i++) {
        if (pos[i] <= 0) {
                pos[i] = 59;
        } else {
                pos[i] = pos[i] - 1;
            }
        }

        /* sleep for 100 milliseconds */
        delay(100);

        /* clears graphic screen */
        cleardevice();
    }

    /* deallocate memory allocated for graphic screen */
    closegraph();
    return 0;
}
```