CGVR PROGRAMS FOR FINAL PRACTICAL **[All The Best A Batch!!]**

1. An architect wishes to visualize the structure of a building/bungalow/kite.Write a program using line drawing algorithm (DDA/Bresenham's), to help him to picture the same.

USING DDA just write the coordinates in the code.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
#include<math.h>
void DDA(float x1,float y1,float x2, float y2)
{
   int i;
   float incx,incy,x0,y0,dx,dy,steps;
   i=1;
   dx=abs(x2-x1);
   dy=abs(y2-y1);
   if(dx>=dy)
      steps=dx;
   else
      steps=dy;
   incx=dx/steps;
   incy=dy/steps;
   x0=x1;
   y0=y1;
   if(x2-x1<0)
   {
      for(i=1;i<=steps;i++)
      {
         putpixel(x0,y0,WHITE);
         x0-=incx;
         y0+=incy;
      }
   }
   else
```

```
   {
        for(i=1;i<=steps;i++)
      {
        putpixel(x0,y0,WHITE);
        x0+=incx;
        y0+=incy;
      }
   }
}
int main()
{
   int gd,gm;
   float n,x,y,h,k;
   gd=DETECT;
   initgraph(&gd,&gm,"c:\\turboc3\\bgi");
    printf("\n\nEnter the value of starting coordinates x and y : ");
   scanf("%f%f",&x,&y);
   printf("\nheight and width required for coordinates of the :::KITE \n");
   scanf("%f%f",&h,&k);
   n=2*h;
   DDA(x,y,x-k,y+h);
   DDA(x,y,x+k,y+h);
   DDA(x+k,y+h,x,y+n);
   DDA(x,y,x,y+n);

   DDA(x-k,y+h,x,y+n);
   DDA(x-k,y+h,x+k,y+h);
   getch();
}
```

KITE USING Bresenham's Algo

```
#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>
int drawline( int x1,int y1,int x2, int y2)
{
```

```
int p ,ax,ay, temp, flag=0;
int x=abs(x2-x1);
int y=abs(y2-y1);
int signx=x2>x1?1:-1;
int signy=y2>y1?1:-1;
if(y>x)
{
temp=x;
x=y;
y=temp;
flag=1;
}
p=(2*y)-x;
putpixel(x1,y1,WHITE);
int counts;
counts=x;
ax=x1;
ay=y1;
while(counts!=0)
{
if(flag==0)
{
if(p<0)
{
ax=ax+signx;
ay=ay+0;
putpixel(ax,ay,WHITE);
p=p+(2*y);
}
else{
ax=ax+signx;
ay=ay+signy;
putpixel(ax,ay,WHITE);
p=p+(2*y)-(2*x);
}
}
```

```
        else
        {
        if(p<0)
        {
        ax=ax+0;
        ay=ay+signy;
        putpixel(ax,ay,WHITE);
        p=p+(2*y);
        }
        else{
        ax=ax+signx;
        ay=ay+signy;
        putpixel(ax,ay,WHITE);
        p=p+(2*y)-(2*x);
        }
        }
        counts--;
        }
return(0);
}
int main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\turboc3\\bgi");
          //  drawline(0,255,650,255);
        //drawline(325,0,325,500);
        //kite
        drawline(200,100,300,300);
        drawline(300,300,400,100);
        drawline(400,100,300,50);
        drawline(300,50,200,100);
        drawline(300,300,350,350);
        drawline(300,300,250,350);
        drawline(250,350,350,350);
        getch();
        closegraph();
```

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

2. Draw an hour glass shape/vase/∝/ Ь/ε/ω/ς/~/r/m/?/s / क/ द/ ड/ ट/ ठ figure.

```
#include<stdio.h>          //PROGRAM REMAINS SAME ONLY COORDINATES WILL CHANGE AT RUN TIME
#include<conio.h>
#include<graphics.h>
#include<math.h>
long fact(int n)
{
        if(n==0)
                return 1;
        else
                return n*fact(n-1);
}
long cal(int n, int i)
{
        long cal=fact(n)/(fact(i)*fact(n-i));
        return cal;
}
int main()
{
        int gd=DETECT, gm, px=0, py=0, n, i;
        double u=0.00;
        char a;
        int x[20], y[20];
   initwindow(1000,1000);
        //initgraph(&gd,&gm,"C://TURBOC3/BGI");
        do
        {
                printf("Enter number of points:");
                scanf("%d",&n);
                for(i=0;i<n;i++)
                {
                        printf("Enter co-Ordinates of point %d:",(i+1));
                        scanf("%d %d",&x[i],&y[i]);
                }
                for(i=0;i<n;i++)
                        putpixel(x[i],y[i],WHITE);
```

```
        for(u=0.0;u<=1;u+=0.001)
        {
                px=x[0]*pow((1-u),n);
                py=y[0]*pow((1-u),n);
                for(int j=1;j<n;j++)
                {
                        px=px+cal(n,j)*x[j]*pow(u,j)*pow((1-u),(n-j));
                        py=py+cal(n,j)*y[j]*pow(u,j)*pow((1-u),(n-j));
                }
                px=px+x[n-1]*pow(u,n);
                py=py+y[n-1]*pow(u,n);
                putpixel(px,py,WHITE);
                px=0;
                py=0;
        }
        printf("Continue(y/n)?");
        scanf("%s",&a);
}while(a!='n');
getch();
```

3.Draw two concentric circle using midpoint circle drawing algorithm/bresenham's circle drawing algorithm.

MIDPOINT CIRCLE

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void inner(int,int,int);
void outer(int,int,int);
void main()
{
int gd=DETECT,gm;

clrscr();
initgraph(&gd,&gm,"c:\\tc\\");
inner(100,300,300);
outer(130,300,300);
getch();
}
void inner(int r,int xc,int yc)
{float d;
int x,y;
d=1.25-r;
```

```
x=0;
y=r;
do
{
if(d<0)
{

x=x+1;
d=d+2*x+1;
}
else
{
x=x+1;
y=y-1;
d=d+2*x-2*y+10;}
putpixel(xc+x,yc+y,5);
putpixel(xc-y,yc-x,5);
putpixel(xc+y,yc-x,5);
putpixel(xc-y,yc+x,5);
putpixel(xc+y,yc+x,5);
putpixel(xc-x,yc-y,5);
putpixel(xc+x,yc-y,5);
putpixel(xc-x,yc+y,5);
}
while(x<y);
}
void outer(int r,int xc,int yc)
{float d;
int x,y;
d=1.25-r;
x=0;
y=r;
do
{
if(d<0)
{
x=x+6;
d=d+2*x+1;
}
else
{
x=x+6;
y=y-6;
d=d+2*x-2*y+10;
}
```

```
 putpixel(xc+x,yc+y,5);
putpixel(xc-y,yc-x,5);
putpixel(xc+y,yc-x,5);
putpixel(xc-y,yc+x,5);
putpixel(xc+y,yc+x,5);
putpixel(xc-x,yc-y,5);
putpixel(xc+x,yc-y,5);
putpixel(xc-x,yc+y,5);
}
while(x<y);
}
```

Bresenham algorithm

```
#include<graphics.h>
#include<conio.h>
#include<iostream.h>
#include<dos.h>
void circle(int x,int y,int X,int Y)
{
        putpixel(x+X,y+Y,WHITE);
        putpixel((-x)+X,y+Y,WHITE);
        putpixel(x+X,(-y)+Y,WHITE);
        putpixel((-x)+X,(-y)+Y,WHITE);
        putpixel(y+Y,x+X,WHITE);
        putpixel((-y)+Y,x+X,WHITE);
        putpixel(y+Y,(-x)+X,WHITE);
        putpixel((-y)+Y,(-x)+X,WHITE);
}
void plot2(int x0,int y0,int r)
{
        int x,y,p;
        x=0;
        y=r;
        p=3-2*r;                    //initial p
        putpixel(x+x0,y+y0,WHITE);
        while(x<y)
        {       x=x+1;
                if(p<0)
                {       y=y;
                        circle(x,y,x0,y0);
                        p=p+4*x+6;
                }
                else
                {
```

```
                        y=y-1;
                        circle(x,y,x0,y0);
                        p=p+4*(x-y)+10;
                }
        }
 }
void plot(int X,int Y,int r)
{
        int x=0;
        int y=r;
        int c=0,d=0;
        int p=3-(2*r);
        delay(2);
        putpixel(x+X,y+Y,WHITE);
        while(x<y)
        {
          c++;
          x=x+1;
          if(p<0)
          {
                if(c<=50)
                {       circle(x,y,X,Y);
                        p=p+(4*x)+6;
                }
                else
                {
                        if(d<=30)
                                d++;
                        else
                        {
                                d=0;
                                c=0;}
                        p=p+(4*x)+6;
                }
          }
          else
          {
                y=y-1;
                if(c<=50)
                {
                        circle(x,y,X,Y);
                        p=p+(4*(x-y))+10;
                }
```

```
                else
                {
                        if(d<=30)
                        {       d++;
                        }
                        else
                    {   d=0;
                                c=0;    }
                        p=p+(4*(x-y))+10;
                }
            }
            c++;
        }
}
void main()
{
   int a,b,r1,r2;
   int gd = DETECT,gm;
   initgraph(&gd,&gm,"C://TURBOC3//bgi");
   cout<<"Enter the co-ordinates of center";
   cin>>a;
   cin>>b;
   cout<<"Enter the first radius";
   cin>>r1;
   cout<<"Enter the second radius";
   cin>>r2;
   plot2(a,b,r1);
   plot(a,b,r2);
   getch();
}
```

3. Perform transformation (Translate, Rotate, Scale, Reflect, Shear) on a boat/kite/house shaped figure **(If any three work then also it is fine!)**

BOAT: double a[3][7]={{10,15,45,50,40,40,10},{30,15,15,30,45,30,30},{1,1,1,1,1,1,1}}

KITE:double a[3][18]={{100,50,100,150,50,100,150,100,100,100,50,150,100,70,100,130,70,13,0},{250,180,250,180,180,100,180,100,250,100,180,180,100,70,100,70,70,70},{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}};

House:int fig[3][5] = {{100, 50, 50,150,150}, {160,110, 50, 50,110},{ 1, 1, 1, 1, 1}};

float ptmatr[2][20]=**// Just put here the coordinates and change the matrix size!!!**

```
void printpixel(float x,float y )
{
        putpixel(x+320,240-y,WHITE);
```

```
}
void drawline(float x1,float y1,float x2,float y2)
{
	float dx,dy,signx=1,signy=1,p,x,y,temp,l,flag = 0;
	dx = abs(x2-x1);
	dy = abs(y2-y1);
	if((x2-x1)<0)
	{
		signx=-1;
	}
	if((y2-y1)<0)
	{
		signy=-1;
	}
	if(dx<dy)
	{
	    temp = dx;
	    dx = dy;
	    dy = temp;
	    flag=1;
	}
	p =( 2 * dy ) - dx;
	x = x1;
	y = y1;
	printpixel(x,y);
	l=dx;
	while(l>0)
	{
		if(p<0)
		{
			p = p + (2*dy);
			if(flag==1)
			{
				y=y+signy;
				printpixel(x,y);
			}
			else
			{	x=x+signx;
				printpixel(x,y);
			}
		}
		else
		{
```

```
                        p = p + (2 * dy) - (2 * dx);
                        x=x+signx;
                        y=y+signy;
                        printpixel(x,y);
                }
                l--;
        }
}
void matmul(float a[3][3], float b[2][20])
 {
        int i,j,k;
        float c[2][20], sum=0;
        int p=0,q=0;
        for(i=0; i<2; i++)
        {
                for(j=0; j<20; j++)
                {
                        for(k=0; k<3; k++)
                        {
                                sum+=a[i][k]*b[k][j];
                        }
                        c[i][j]=sum;
                        sum=0;
                }
  }
        for (p=0; p<19; p++)
        drawline(c[q][p], c[q+1][p], c[q][p+1], c[q+1][p+1]);
        drawline(c[0][18], c[1][18], c[0][0], c[1][0]);

}
void translate()
{
int tx,ty;
setcolor(WHITE);// sets the color of string
outtextxy(240,10,"TRANSLATION");//displays the text at specified point
outtextxy(238,20,"------------");
printf("\nEnter tx: ");
scanf("%d",&tx);
printf("\nEnter ty: ");
scanf("%d",&ty);
cleardevice();//clears the screen and sets the position back to (0,0)
   drawline(-320,0,320,0);
        drawline(0,-240,0,240);
```

```
            drawline(70,100,170,100);
            drawline(99,130,140,130);
            drawline(70,100,99,130);
            drawline(140,130,170,100);
            drawline(70,100,120,0);
            drawline(170,100,120,0);
            drawline(114,130,85,100);
            drawline(125,130,155,100);
            drawline(85,100,120,0);
            drawline(155,100,120,0);
printf("\nAfter Translation");
   drawline(-320,0,320,0);
            drawline(0,-240,0,240);
            drawline(70+tx,100+ty,170+tx,100+ty);
            drawline(99+tx,130+ty,140+tx,130+ty);
            drawline(70+tx,100+ty,99+tx,130+ty);
            drawline(140+tx,130+ty,170+tx,100+ty);
            drawline(70+tx,100+ty,120+tx,0+ty);
            drawline(170+tx,100+ty,120+tx,0+ty);
            drawline(114+tx,130+ty,85+tx,100+ty);
            drawline(125+tx,130+ty,155+tx,100+ty);
            drawline(85+tx,100+ty,120+tx,0+ty);
            drawline(155+tx,100+ty,120+tx,0+ty);
}
void scale()
{
int sx,sy;
setcolor(WHITE);
outtextxy(240,10,"SCALING");
outtextxy(238,20,"--------");
printf("\nEnter sx: ");
scanf("%d",&sx);
printf("\nEnter sy: ");
scanf("%d",&sy);
cleardevice();
   drawline(-320,0,320,0);
   drawline(0,-240,0,240);
            drawline(70,100,170,100);
            drawline(99,130,140,130);
            drawline(70,100,99,130);
            drawline(140,130,170,100);
            drawline(70,100,120,0);
            drawline(170,100,120,0);
```

```
        drawline(114,130,85,100);
        drawline(125,130,155,100);
        drawline(85,100,120,0);
        drawline(155,100,120,0);
printf("\nAfter Scaling");
        drawline(-320,0,320,0);
   drawline(0,-240,0,240);
        drawline(70*sx,100*sy,170*sx,100*sy);
        drawline(99*sx,130*sy,140*sx,130*sy);
        drawline(70*sx,100*sy,99*sx,130*sy);
        drawline(140*sx,130*sy,170*sx,100*sy);
        drawline(70*sx,100*sy,120*sx,0*sy);
        drawline(170*sx,100*sy,120*sx,0*sy);
        drawline(114*sx,130*sy,85*sx,100*sy);
        drawline(125*sx,130*sy,155*sx,100*sy);
        drawline(85*sx,100*sy,120*sx,0*sy);
        drawline(155*sx,100*sy,120*sx,0*sy);
}
void shear()
 {
        float shx, shy, sx1,sy1,shearmatrix[3][3]={{1,0,0}, {0,1,0}, {0,0,1}};
        printf("Enter shearing parameters shx& shy: ");
        scanf("%f%f",&shx,&shy);
        cleardevice();
        shearmatrix[0][1]=shx;
        shearmatrix[1][0]=shy;
        sx1=shearmatrix[0][1];
        sy1=shearmatrix[1][0];
        drawline(-320,0,320,0);
        drawline(0,-240,0,240);
        drawline(70+sx1*sy1,100*sy1,99+sx1*sy1,130*sy1);
        drawline(99+sx1*sy1,130*sy1,140+sx1*sy1,130*sy1);
        drawline(70+sx1*sy1,100*sy1,99+sx1*sy1,130*sy1);
        drawline(140+sx1*sy1,130*sy1,170+sx1*sy1,100*sy1);
        drawline(70+sx1*sy1,100*sy1,120+sx1*sy1,0*sy1);
        drawline(170+sx1*sy1,100*sy1,120+sx1*sy1,0*sy1);
        drawline(114+sx1*sy1,130*sy1,85+sx1*sy1,100*sy1);
        drawline(125+sx1*sy1,130*sy1,155+sx1*sy1,100*sy1);
        drawline(85+sx1*sy1,100*sy1,120+sx1*sy1,0*sy1);
        drawline(155+sx1*sy1,100*sy1,120+sx1*sy1,0*sy1);
}

void rotate()
```

```
{
float deg,rotnmatrix[3][3]={{0,0,0}, {0,0,0}, {0,0,1}};
        printf("\n\tEnter value of angle in degree :-\t");
        scanf("%f",&deg);
        rotnmatrix[0][0]=cos(deg*(3.142857/180));
        rotnmatrix[0][1]=-sin(deg*(3.142857/180));
        rotnmatrix[1][0]=sin(deg*(3.142857/180));
        rotnmatrix[1][1]=cos(deg*(3.142857/180));
        matmul(rotnmatrix, ptmatr);
}
void refln()
{
        float flag, sx0,sy0,reflnmatrix[3][3]={0,0,0,0,0,0,0,0,1};
        printf("\n\tEnter '0' for reflection on X axis or \n\t'1' for reflection on Y axis or \n\t'2' for reflection about
origin :-\t");
        scanf("%f",&flag);
        if(flag==0)
        {
                reflnmatrix[0][0]=1;
                reflnmatrix[1][1]=-1;
        }
else

        {
                if(flag==1)
                {
                        reflnmatrix[0][0]=-1;
                        reflnmatrix[1][1]=1;
                }
                else
                {
                        reflnmatrix[0][0]=-1;
                        reflnmatrix[1][1]=-1;
                }
        }
        //matmul(reflnmatrix, ptmatr);
        sx0=reflnmatrix[0][0];
        sy0=reflnmatrix[1][1];
        drawline(-320,0,320,0);
        drawline(0,-240,0,240);
        drawline(70*sx0,100*sy0,170*sx0,100*sy0);
        drawline(99*sx0,130*sy0,140*sx0,130*sy0);
        drawline(70*sx0,100*sy0,99*sx0,130*sy0);
```

```
            drawline(140*sx0,130*sy0,170*sx0,100*sy0);
            drawline(70*sx0,100*sy0,120*sx0,0*sy0);
            drawline(170*sx0,100*sy0,120*sx0,0*sy0);
            drawline(114*sx0,130*sy0,85*sx0,100*sy0);
            drawline(125*sx0,130*sy0,155*sx0,100*sy0);
            drawline(85*sx0,100*sy0,120*sx0,0*sy0);
            drawline(155*sx0,100*sy0,120*sx0,0*sy0);
}
int main()
{
             int ch;
            int gd = DETECT , gm;
            initgraph(&gd,&gm,"c:\\turboc3\\bgi");
            drawline(-320,0,320,0);
            drawline(0,-240,0,240);
             printf("---2D TRANSFORMATIONS---");
printf("\n1)Translate\n2)scaing\n3)Rotation\n\n4)Shearing\n5)Reflection\n");
printf("\nEnter your choice: ");
scanf("%d",&ch);
cleardevice();
switch(ch)
{
case 1: translate();
break;
case 2: scale();
break;
case 3: rotate();
break;
case 4: shear();
break;
case 5: refln();
break;
default: printf("you have enterd wrong choice");
break;
}
            getch();
            return 0;
}
```

4. Write a program to implement Cohen Sutherland algorithm/Liang Barsky algorithm

Cohen Sutherland

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

```
#include<graphics.h>
int LEFT=1,RIGHT=2,BOTTOM=4,TOP=8,xl,yl,xh,yh;
int getcode(int x,int y){
	int code = 0;
	//Perform Bitwise OR to get outcode
	if(y > yh) code |=TOP;
	if(y < yl) code |=BOTTOM;
	if(x < xl) code |=LEFT;
	if(x > xh) code |=RIGHT;
	return code;
}
void main()
{
	int gdriver = DETECT,gmode;
	initgraph(&gdriver,&gmode,"C:\\TURBOC3\\BGI");
	setcolor(GREEN);
	cout<<"Enter bottom left and top right co-ordinates of window: ";
	cin>>xl>>yl>>xh>>yh;
	rectangle(xl,yl,xh,yh);
	int x1,y1,x2,y2;
	cout<<"Enter the endpoints of the line: ";
	cin>>x1>>y1>>x2>>y2;
	line(x1,y1,x2,y2);
	getch();

	int outcode1=getcode(x1,y1), outcode2=getcode(x2,y2);
	int accept = 0;   //decides if line is to be drawn
	while(1){
		float m =(float)(y2-y1)/(x2-x1);
		//Both points inside. Accept line
		if(outcode1==0 && outcode2==0){
			accept = 1;
			break;
		}
		//AND of both codes != 0.Line is outside. Reject line
		else if((outcode1 & outcode2)!=0){
			break;
		}else{
			int x,y;
			int temp;
			//Decide if point1 is inside, if not, calculate intersection
			if(outcode1==0)
				temp = outcode2;
```

```
                    else
                            temp = outcode1;
                    //Line clips top edge
                    if(temp & TOP){
                            x = x1+ (yh-y1)/m;
                            y = yh;
                    }
                    else if(temp & BOTTOM){          //Line clips bottom edge
                            x = x1+ (yl-y1)/m;
                            y = yl;
                    }else if(temp & LEFT){    //Line clips left edge
                            x = xl;
                            y = y1+ m*(xl-x1);
                    }else if(temp & RIGHT){  //Line clips right edge
                            x = xh;
                            y = y1+ m*(xh-x1);
                    }
                    //Check which point we had selected earlier as temp, and replace its co-ordinates
                    if(temp == outcode1){
                            x1 = x;
                            y1 = y;
                            outcode1 = getcode(x1,y1);
                    }else{
                            x2 = x;
                            y2 = y;
                            outcode2 = getcode(x2,y2);
                    }
            }
    }
    setcolor(WHITE);
    cout<<"After clipping:";
accept)
            line(x1,y1,x2,y2);
    getch();
    closegraph();
}
```

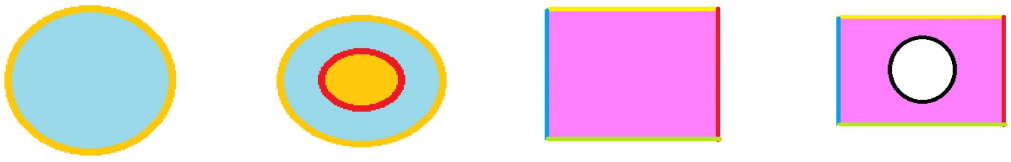
Liang Barsky

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
int cliptest(float p,float q,float *u1,float *u2)
{
```

```
        float r;
        int retval=1;
        if(p<0.0)
        {
           r=q/p;
        if(r>*u2)
           retval=0;
        if(r>*u1)
          *u1=r;
        }
        else if(p>0.0)
        {
           r=q/p;
           if(r<*u1)
           retval=0;
        if(r<*u2)
           *u2=r;
        }
        else if(q<0.0)
           retval=0;
        return(retval);
}
void clipline(int minx,int miny,int maxx,int maxy,int x1,int y1,int x2,int y2)
{
        float u1=0.0,u2=1.0,dx=x2-x1,dy;
        if(cliptest(-dx,x1-minx,&u1,&u2))
        if(cliptest(dx,maxx-x1,&u1,&u2))
          {
          dy=y2-y1;
          if(cliptest(-dy,y1-miny,&u1,&u2))
          if(cliptest(dy,maxy-y1,&u1,&u2))
           {
              if(u2<1.0)
              {
                x2=x1+u2*dx;
                y2=y1+u2*dy;
              }
               if(u1>0.0)
                {
                 x1+=u1*dx;
                 y1+=u1*dy;
                }
                 line(x1,y1,x2,y2);
           } }
}
```

```
void main()
  {
  int gdriver=DETECT,gmode,x1,y1,x2,y2,minx,miny,maxx,maxy;
  initgraph(&gdriver,&gmode,"");
  clrscr();
  printf("ENTER COORDINATES OF FIRST POINT");
  scanf("%d %d",&x1,&y1);
  printf("ENTER COORDINATES OF 2ND POINT");
  scanf("%d %d",&x2,&y2);
  printf("Enter the min & max x values: ");
  scanf("%d %d",&minx,&maxx);
  printf("Enter the min & max y values: ");
  scanf("%d %d",&miny,&maxy);
  clrscr();
  printf("Before clipping");
  line(x1,y1,x2,y2);
  getch();
  clrscr();
  clipline(minx,miny,maxx,maxy,x1, y1, x2, y2);
  rectangle(minx,maxy,maxx,miny);
  getch();
}
```

5. Write a program to implement polygon clipping wrt left/right/top/bottom window edge. (ANY METHOD WILL DO: COHEN OR LIANG)
6. Write a program to visualize the color combination for the hut/boat /kite.
7. Write a program to color the areas



Boundary fill 8 connected for drawing and filling a rectangle and a circle inside

```
#include<stdio.h>
#include<graphics.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);6
        boundaryfill(x-1,y,f_color,b_color);
        boundaryfill(x,y-1,f_color,b_color);
        boundaryfill(x+1,y+1,f_color,b_color);
        boundaryfill(x-1,y-1,f_color,b_color);
        boundaryfill(x+1,y+1,f_color,b_color);
        boundaryfill(x-1,y-1,f_color,b_color);
    }
}
int main()
{
    int gm,gd=DETECT,radius,radius1,radius2;
    int x,y;
   printf("Enter the co-ordintes for center of the circle\n");
    scanf("%d%d",&x,&y);
    printf("Enter radius of circle\n");
    scanf("%d",&radius);
    printf("Enter radius of circle\n");
    scanf("%d",&radius1);
    printf("Enter radius of circle\n");
    scanf("%d",&radius2);
    initgraph(&gd,&gm,"c:\\turboc3\\bgi");
    circle(x,y,radius);
    circle(x,y,radius1);
    circle(x,y,radius2);
    rectangle(70,70,350,370);
    boundaryfill(x,y,LIGHTBLUE,WHITE);
    delay(50);
    closegraph();
    return 0;
}
```

8. Write a program to simulate fish swimming

```
#include<stdio.h>
#include<graphics.h>
#include<dos.h>
#include<conio.h>
#define ScreenWidth getmaxx()
#define ScreenHeight getmaxy()
int ldisp=0;
```

```
void draw_box();
void draw_box()
{
   line(20,100,600,100);
   line(20,100,20,400);
   line(600,100,600,400);
   line(600,400,20,400);
   //pieslice(50,100,500,400);
}
void draw_fish(int x)
{
   ellipse(x,200,0,360,50,30);
   line(x-50,199,x-75,170);
   line(x-50,199,x-75,230);
   line(x-75,170,x-75,230);
   circle(x+15,190,3);
}
void draw_fish2(int x)
{
ellipse(x,200,0,360,50,30);
line(x+50,200,x+75,170);
line(x+50,200,x+75,230);
line(x+75,170,x+75,230);
circle(x-15,190,3);
}
int main()
{
   int gd=DETECT,gm,x=100;
  //Change BGI directory according to yours
  initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
   //draw_box();
 //  system("pause");
   while(x!=500)
  {
     draw_box();
   ldisp=(ldisp+2)%20;
   draw_fish(x);
   delay(30);
   cleardevice();
   x=(x+2)%ScreenWidth;
}
x=500;
while(x!=100)
```

```
 {
    draw_box();
  ldisp=(ldisp+2)%20;
  draw_fish2(x);
  delay(30);
  cleardevice();
  x=(x-2)%ScreenWidth;
}
return(0);
 }
```

// more easy

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<graphics.h>
#include<time.h>
#include<dos.h>

int main()
{
        int gd=DETECT ,gm;
        int x=10, y=200, x1=675, y1=380;
        int stangle=35, endangle=140, radius=90;
        initgraph(&gd,&gm,"c:\\turboc3\\bgi");

        while(!kbhit())
        {

        cleardevice();
        setbkcolor(BLUE);
        if(x<640)
        {
        x=x+5;
        y=y+1;
        arc(x,y,stangle,endangle+35,radius);
        arc(x,y-110,190,323,radius+2);
        circle(x+40,y-60,5);
        }
        /*   else{
```

```
        x=x-5;
        y=y-1;
        arc(x1,y1,stangle-30,endangle+4,radius);
        arc(x1,y1-110,217,350,radius+2);
        circle(x1-40,y1-60,5);

        }*/
        setcolor(YELLOW);
        delay(200);
        }
        closegraph();
}
```

9. A plane taking off at various angles is to be shown to train a set of pilots. Simulate the take off process to help the training.
10. A teacher wants to explain the solar system to school students. Simulate the demonstration.

SOLAR SYSTEM

```
#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);
            cleardevice();
      }
      closegraph();
      return 0;
 }
```

11.Write a program to simulate a car moving on road

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
for (int i=0;i<500;i++)
{
/***CAR BODY ******/
line(50+i,370,90+i,370);
arc(110+i,370,0,180,20);
line(130+i,370,220+i,370);
arc(240+i,370,0,180,20);    //int x, int y, int stangle, int endangle, int radius
line(260+i,370,300+i,370);
line(300+i,370,300+i,350);
line(300+i,350,240+i,330);
line(240+i,330,200+i,300);
line(200+i,300,110+i,300);
line(110+i,300,80+i,330);
line(80+i,330,50+i,340);
```

```
line(50+i,340,50+i,370);
/***CAR Windows***/
line(165+i,305,165+i,330);
line(165+i,330,230+i,330);
line(230+i,330,195+i,305);
line(195+i,305,165+i,305);
line(160+i,305,160+i,330);
line(160+i,330,95+i,330);
line(95+i,330,120+i,305);
line(120+i,305,160+i,305);
/**Wheels**/
circle(110+i,370,17);
circle(240+i,370,17);
delay(10);
cleardevice();
line(0,390,639,390);  //ROAD
}
getch();
}
```

12.Write a program to simulate a train moving on tracks

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

void main()
{
int gd=DETECT,gm;
        int i=0;
        initgraph(&gd,&gm,"..\\bgi");
        while(!kbhit())
        {
                i++;
                cleardevice();
                line(80+i,270,80+i,300);
                line(80+i,300,160+i,300);
                line(160+i,300,160+i,270);
                line(80+i,270,160+i,270);
                line(140+i,270,140+i,245);
                line(140+i,245,160+i,245);
                line(160+i,245,160+i,270);
                line(152+i,245,152+i,235);
```

```
                line(152+i,235,157+i,235);
                line(157+i,235,157+i,245);
                circle(90+i,305,5);
                circle(105+i,305,5);
                circle(120+i,305,5);
                circle(135+i,305,5);
                circle(150+i,305,5);
                line(0,315,950,300);
                delay(10);
        }
        getch();
        closegraph();
}
```

13. Write a program to draw koch curve.

```
#include <graphics.h>
#include <math.h>
#include <conio.h>
#include<stdio.h>
void Koch(float startx, float endx, float starty, float endy, int level){
   float x1, x2, x3, y1, y2, y3, L, cosa, sina, h;
   if(level==1){
        line(startx, starty, endx, endy);
   }
   else{
        L = sqrt( (endx-startx) * (endx-startx) + (endy-starty) * (endy-starty) );
        h = L /(2 * sqrt(3));
        sina = (endy - starty)/L;
        cosa = (endx - startx)/L;
        x1 = startx + (endx - startx)/3;
        x2 = (endx + startx)/2 + h * sina;
        x3 = startx + 2 * (endx - startx)/3;

        y1 = starty + (endy - starty)/3;
        y2 = (endy + starty)/2 - h * cosa;
        y3 = starty + 2 * (endy - starty)/3;

        Koch(startx, x1, starty, y1, level-1);
        Koch(x1, x2, y1, y2, level-1);
        Koch(x2, x3, y2, y3, level-1);
        Koch(x3, endx, y3, endy, level-1);
   }
}
int main(){
   int Gdriver, Gmode;
   float x1, x2, y1, y2;
   Gdriver = DETECT;
   //Gmode ;
   initgraph(&Gdriver, &Gmode, "C://Turboc3//BGI");
   x1 = 0;
```

```
   x2 = getmaxx();
   y1 = getmaxy();
   y2 = y1;
   Koch(x1, x2, y1, y2, 6);
   getch();
   closegraph();return(0):
}
```

14.Write a program to create a fractal tree(TOO COMPLEX!)

```
#define _WIN32_WINNT 0x0500
#include<windows.h>
#include <string>
#include <math.h>
using namespace std;
const float PI = 3.1415926536f;
class myBitmap
{
public:
   myBitmap() : pen( NULL ) {}
   ~myBitmap()
   {
          DeleteObject( pen );
          DeleteDC( hdc );
          DeleteObject( bmp );
   }
   bool create( int w, int h )
   {
          BITMAPINFO     bi;
          void                *pBits;
          ZeroMemory( &bi, sizeof( bi ) );
          bi.bmiHeader.biSize         = sizeof( bi.bmiHeader );
          bi.bmiHeader.biBitCount             = sizeof( DWORD ) * 8;
          bi.bmiHeader.biCompression = BI_RGB;
          bi.bmiHeader.biPlanes      = 1;
          bi.bmiHeader.biWidth       =  w;
          bi.bmiHeader.biHeight      = -h;

          HDC dc = GetDC( GetConsoleWindow() );
          bmp = CreateDIBSection( dc, &bi, DIB_RGB_COLORS, &pBits, NULL, 0 );
          if( !bmp ) return false;

          hdc = CreateCompatibleDC( dc );
          SelectObject( hdc, bmp );
          ReleaseDC( GetConsoleWindow(), dc );
```

```
	width = w; height = h;

	return true;
}

void setPenColor( DWORD clr )
{
	if( pen ) DeleteObject( pen );
	pen = CreatePen( PS_SOLID, 1, clr );
	SelectObject( hdc, pen );
}

void saveBitmap( string path )
{
	BITMAPFILEHEADER	fileheader;
	BITMAPINFO		infoheader;
	BITMAP			bitmap;
	DWORD*			dwpBits;
	DWORD			wb;
	HANDLE			file;

	GetObject( bmp, sizeof( bitmap ), &bitmap );

	dwpBits = new DWORD[bitmap.bmWidth * bitmap.bmHeight];
	ZeroMemory( dwpBits, bitmap.bmWidth * bitmap.bmHeight * sizeof( DWORD ) );
	ZeroMemory( &infoheader, sizeof( BITMAPINFO ) );
	ZeroMemory( &fileheader, sizeof( BITMAPFILEHEADER ) );

	infoheader.bmiHeader.biBitCount = sizeof( DWORD ) * 8;
	infoheader.bmiHeader.biCompression = BI_RGB;
	infoheader.bmiHeader.biPlanes = 1;
	infoheader.bmiHeader.biSize = sizeof( infoheader.bmiHeader );
	infoheader.bmiHeader.biHeight = bitmap.bmHeight;
	infoheader.bmiHeader.biWidth = bitmap.bmWidth;
	infoheader.bmiHeader.biSizeImage = bitmap.bmWidth * bitmap.bmHeight * sizeof( DWORD );

	fileheader.bfType    = 0x4D42;
	fileheader.bfOffBits = sizeof( infoheader.bmiHeader ) + sizeof( BITMAPFILEHEADER );
	fileheader.bfSize    = fileheader.bfOffBits + infoheader.bmiHeader.biSizeImage;

	GetDIBits( hdc, bmp, 0, height, ( LPVOID )dwpBits, &infoheader, DIB_RGB_COLORS );
```

```
        file = CreateFile( path.c_str(), GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL
);
        WriteFile( file, &fileheader, sizeof( BITMAPFILEHEADER ), &wb, NULL );
        WriteFile( file, &infoheader.bmiHeader, sizeof( infoheader.bmiHeader ), &wb, NULL );
        WriteFile( file, dwpBits, bitmap.bmWidth * bitmap.bmHeight * 4, &wb, NULL );
        CloseHandle( file );

        delete [] dwpBits;
   }

   HDC getDC()    { return hdc; }
   int getWidth()  { return width; }
   int getHeight() { return height; }

private:
   HBITMAP bmp;
   HDC      hdc;
   HPEN    pen;
   int     width, height;
};
class vector2
{
public:
   vector2() { x = y = 0; }
   vector2( int a, int b ) { x = a; y = b; }
   void set( int a, int b ) { x = a; y = b; }
   void rotate( float angle_r )
   {
        float _x = static_cast<float>( x ),
             _y = static_cast<float>( y ),
              s = sinf( angle_r ),
              c = cosf( angle_r ),
              a = _x * c - _y * s,
              b = _x * s + _y * c;

        x = static_cast<int>( a );
        y = static_cast<int>( b );
   }

   int x, y;
};
class fractalTree
{
```

```
public:
   fractalTree()                    { _ang = DegToRadian( 24.0f ); }
   float DegToRadian( float degree ) { return degree * ( PI / 180.0f ); }

   void create( myBitmap* bmp )
   {
         _bmp = bmp;
         float line_len = 130.0f;

         vector2 sp( _bmp->getWidth() / 2, _bmp->getHeight() - 1 );
         MoveToEx( _bmp->getDC(), sp.x, sp.y, NULL );
         sp.y -= static_cast<int>( line_len );
         LineTo( _bmp->getDC(), sp.x, sp.y);

         drawRL( &sp, line_len, 0, true );
         drawRL( &sp, line_len, 0, false );
   }

private:
   void drawRL( vector2* sp, float line_len, float a, bool rg )
   {
         line_len *= .75f;
         if( line_len < 2.0f ) return;

         MoveToEx( _bmp->getDC(), sp->x, sp->y, NULL );
         vector2 r( 0, static_cast<int>( line_len ) );

      if( rg ) a -= _ang;
      else a += _ang;

         r.rotate( a );
         r.x += sp->x; r.y = sp->y - r.y;

         LineTo( _bmp->getDC(), r.x, r.y );

         drawRL( &r, line_len, a, true );
         drawRL( &r, line_len, a, false );
   }

   myBitmap* _bmp;
   float     _ang;
};
int main( int argc, char* argv[] )
```

```
{
   ShowWindow( GetConsoleWindow(), SW_MAXIMIZE );

   myBitmap bmp;
   bmp.create( 640, 512 );
   bmp.setPenColor( RGB( 255, 255, 0 ) );

   fractalTree tree;
   tree.create( &bmp );

   BitBlt( GetDC( GetConsoleWindow() ), 0, 20, 648, 512, bmp.getDC(), 0, 0, SRCCOPY );

   bmp.saveBitmap( "f://rc//fracTree.bmp" );

   system( "pause" );

   return 0;
}
```

15. Write a program to create a fern leaf

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
int a;
void drawfern(int x,int y,int l,int arg,int n)
{
int x1,y1,i;
int l1,xpt,ypt;
if(n>0)
 {
 x1=(int)(x-l*sin(arg*3.14/180));
 y1=(int)(y-l*cos(arg*3.14/180));
 line(x,y,x1,y1);
 l1=(int)(l/5);
 for(i=1;i<6;i++)
  {
  xpt=(int)(x-i*l1*sin(arg*3.14/180));
  ypt=(int)(y-i*l1*cos(arg*3.14/180));
  drawfern(xpt,ypt,(int)(l/(i+1)),arg+a,n-1);
  drawfern(xpt,ypt,(int)(l/(i+1)),arg-a,n-1);
  }
 }
```

```
}
int main()
{
int gd=DETECT,gm,x,y,l;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI\\");
x=getmaxx()/2;
y=getmaxy()/2;
l=150;
a=45;
setcolor(YELLOW);
drawfern(x,y,l,0,5);
getch();
return(0);
}
```

16.Write a program to demonstrate Sierpinski Triangle

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void drawtriangle(int x1, int y1, int x2, int y2, int x3, int y3)
{
   if(x1-x2 > 4)
   {
      line(x1,y1,x2,y2);
      line(x2,y2,x3,y3);
      line(x3,y3,x1,y1);
      drawtriangle(x1,y1,(x1+x2)/2, (y1+y2)/2, (x1+x3)/2, (y1+y3)/2);
      drawtriangle((x1+x2)/2, (y1+y2)/2,x2,y2, (x2+x3)/2, (y2+y3)/2);
      drawtriangle((x1+x3)/2, (y1+y3)/2,(x2+x3)/2, (y2+y3)/2, x3, y3);
   }
   return;
}
int main()
{
   int gd=DETECT, gm;
   initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
   drawtriangle(getmaxx()/2, 0, 0, getmaxy(), getmaxx(), getmaxy());
   getch();
   closegraph();
   return(0);
}
```

18.Write a program to rotate a kite/hut/boat about a given point.

Boat (For kite and hut logic is same only coordinates will change)

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
#include <math.h>
#define PI 3.14159265
double ans[3][7];
float a[3][7]={{10,15,45,50,40,40,10},{30,15,15,30,45,30,30},{1,1,1,1,1,1,1}};

void printpixel(float x,float y )
{
	putpixel(x+320,240-y,WHITE);
}

void drawline(float x1,float y1,float x2,float y2)
{
	float dx,dy,signx=1,signy=1,p,x,y,temp,l,flag = 0;
	dx = abs(x2-x1);
	dy = abs(y2-y1);
	if((x2-x1)<0)
	{
		signx=-1;
	}
	if((y2-y1)<0)
	{
		signy=-1;
	}
	if(dx<dy)
	{
	    temp = dx;
	    dx = dy;
	    dy = temp;
	    flag=1;
	}
	p =( 2 * dy ) - dx;
	x = x1;
	y = y1;
	printpixel(x,y);
	l=dx;
	while(l>0)
	{
		if(p<0)
		{
			p = p + (2*dy);
			if(flag==1)
			{
				y=y+signy;
				printpixel(x,y);
```

```
                        }
                        else
                        {       x=x+signx;
                                printpixel(x,y);
                        }
                }
                else
                {
                        p = p + (2 * dy) - (2 * dx);
                        x=x+signx;
                        y=y+signy;
                        printpixel(x,y);
                }
                l--;
        }

}
void matmul(double b[3][3])
 {

        int i,j,k;
        float c[3][7], sum=0;
        int p=0,q=0;
        for(i=0;i<3;i++)
        {
        for(j=0;j<7;j++)
                ans[i][j]=0;
        }
        for(i=0; i<3; i++)
        {
                for(j=0; j<7; j++)
                {
                        for(k=0; k<3; k++)
                        {
                                ans[i][j]+=b[i][k]*a[k][j];
                        }
                        c[i][j]=sum;
                        sum=0;

                }
  }
  for(i=0;i<3;i++)
        {
        for(j=0;j<7;j++)
                a[i][j]= ans[i][j];

        }
```

```
            for (p=0; p<6; p++)
            drawline(c[q][p], c[q+1][p], c[q][p+1], c[q+1][p+1]);


}
void rotation(double t)
{
            double x, cost,sint, val;
            x = t;
            val = PI / 180.0;
            cost = cos( x*val );
            sint= sin( x*val );
            double b[3][3]={{cost,-sint,0},{sint,cost,0},{0,0,1}};

            matmul(b);

}
void translation(double tx,double ty)
{

            double b[3][3]={{1,0,tx},{0,1,ty},{0,0,1}};
            matmul(b);

}

int main()
{
            double x1,y1,x2,y2;
   int tx,ty;
            int i,j;
            int X,Y;
             int ch;
            int gd = DETECT , gm;
            initgraph(&gd,&gm,"c:\\turboc3\\bgi");

            tx=a[0][0];
            ty=a[1][0];
            translation(-tx,-ty);
            rotation(30);
            translation(tx,ty);
            //rotated matrix
            for(i=0;i<6;i++)
    {

             drawline(x1+ans[0][i],y1-ans[1][i],x1+ans[0][i+1],y1-ans[1][i+1]);
    }
            getch();
            return 0;
```

}