MAHARAJA SURAJMAL INSTITUTE

COMPUTER GRAPHICS

(BCA 303)

Submitted By: SOUMYA AWASTHI **Submitted to:** MR. KUMAR GAURAV

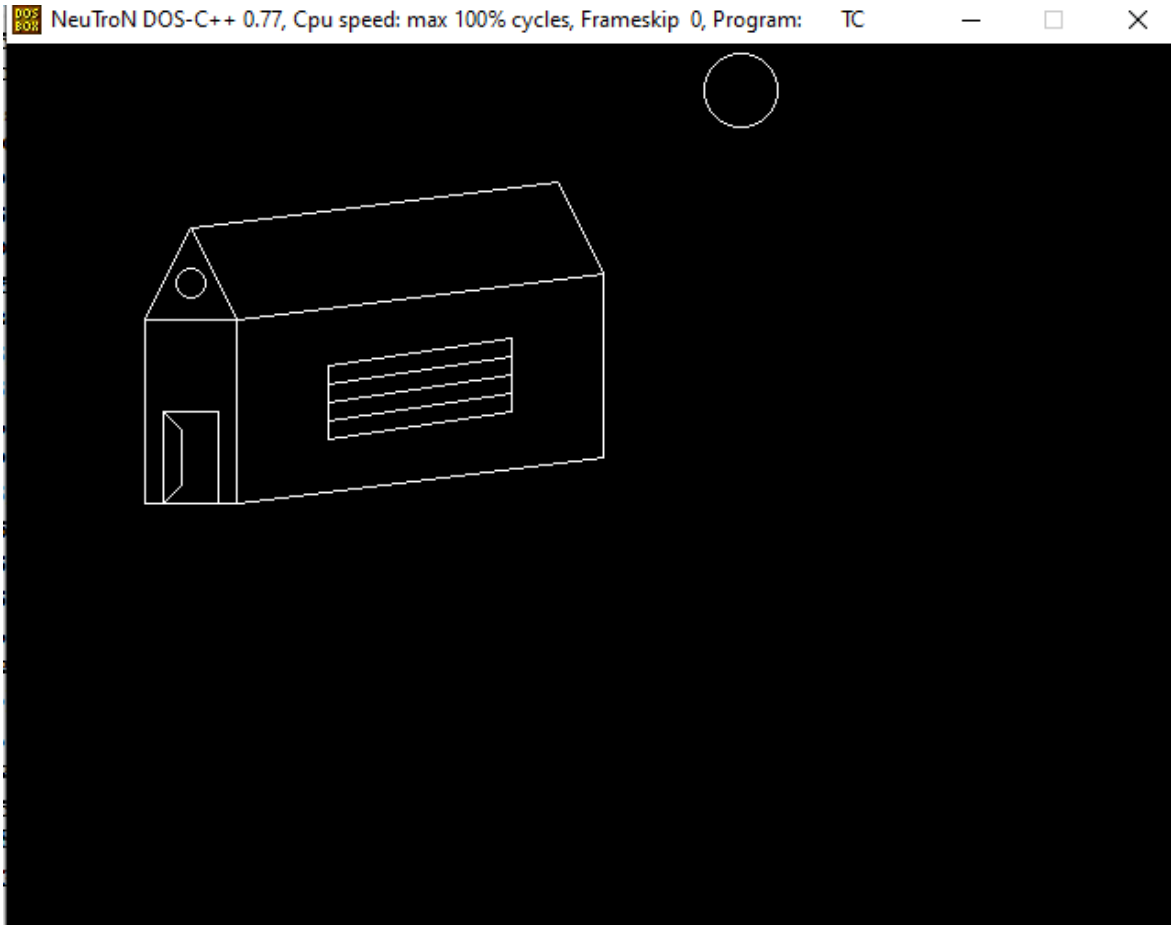
ROLL NUMBER : 08114902018
COURSE : BCA 5TH SEMESTER
SECTION : ‘ B ’
SHIFT : Morning

INDEX

S.NO	PROGRAM	Signature
1.	WAP to Draw Objects (HUT/SMILY Face)	
2.	WAP to draw a line using DDA Line Drawing Algorithm.	
3.	WAP to Draw a line using Bresenham's Line Drawing Algorithm.	
4.	WAP to Draw a line using Mid-Point Line Drawing Algorithm.	
5.	WAP to draw a circle using Bresenham's Circle Drawing Algorithm.	
6.	WAP to draw a circle using Mid-Point Circle Drawing Algorithm.	
7.	WAP to draw a Ellipse using Mid-point Ellipse Drawing Algorithm.	
8.	WAP for Translation in 2-Dimension.	
9.	WAP for Translation in 2-Dimension (Circle).	
10.	**WAP To Rotate A Rectangle At One Of Its Coordinate In Clockwise Direction.**	
11.	**WAP To Rotate A Rectangle At One Of Its Coordinate In Anticlockwise Direction.**	
12.	**WAP To Scale A Square To Double Its Size.**	
13.	WAP for Reflection in 2-Dimension.	
14.	WAP to Shear a Rectangle in X- direction (2-Dimension.)	
15.	WAP to Shear a Rectangle in Y- direction (2-Dimension.)	
16.	WAP to Shear a Rectangle in X and Y- direction (2-Dimension.)	
17.	WAP to Draw a Rectangle Using DDA Algorithm.	
18.	WAP to rotate a coin on table.	
19.	WAP for Cohen Sutherland's Algorithm	
19.	WAP to draw flying balloons	
20.	WAP to rotate a circle outside another circle.	
21.	WAP to rotate a circle inside and outside another circle alternatively.	
22.	WAP to changing radius of circle.	
23.	**WAP To a Wheel.**	
24.	**WAP To Rotate the Triangle at Its Centre In Clockwise Direction**	
25.	**WAP To Change the Triangle in To A Rectangle.**	
26.	WAP to draw Analog clock.	

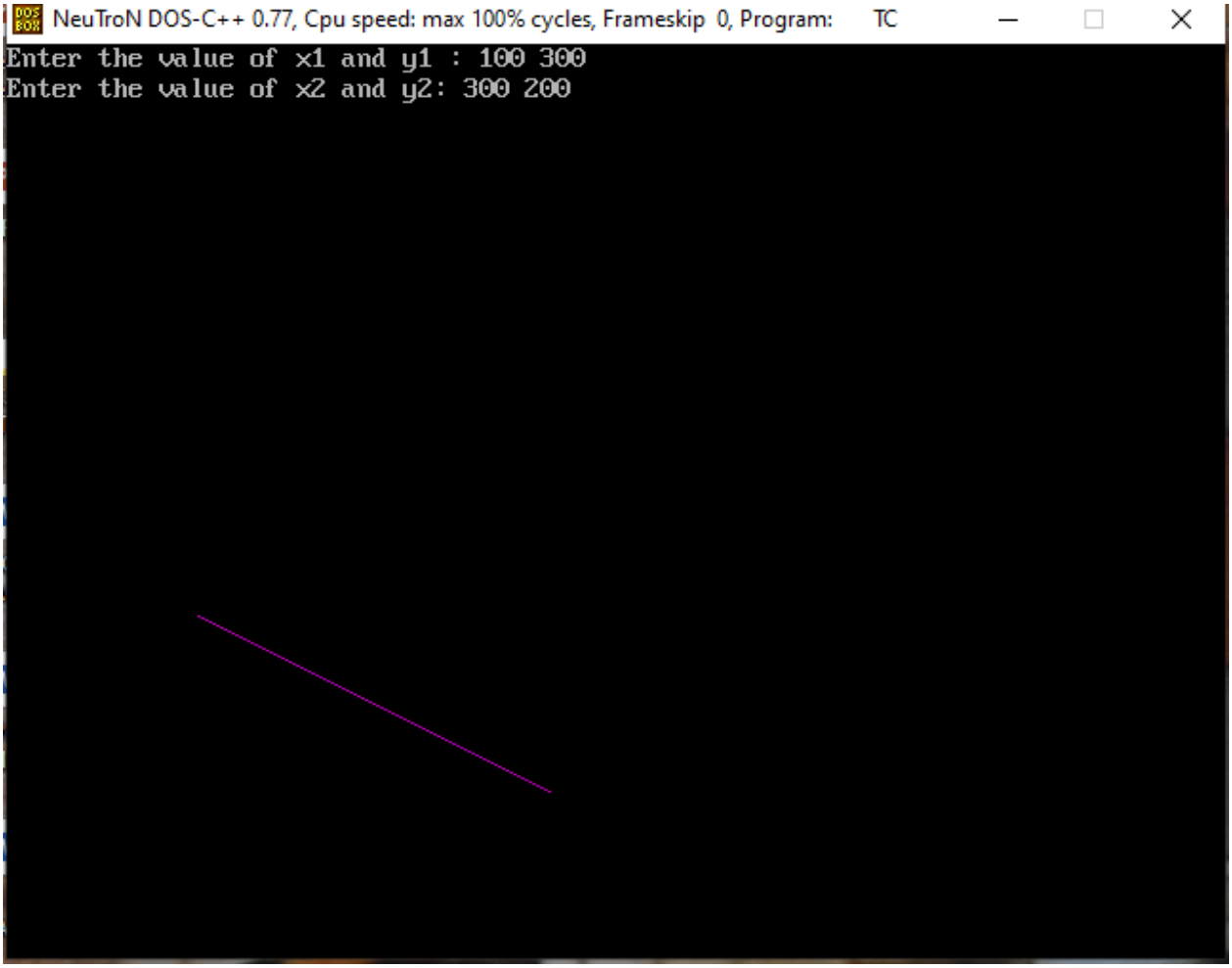
Q.1 WAP to Draw Objects (HUT/SMILY Face)

```
#include "stdio.h"
#include "conio.h"
#include "graphics.h"
void main()
{
 int gd=DETECT,gm;
 initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
 line(100,100,75,150);
 line(100,100,125,150);
 moveto(75,150);
 lineto(125,150);
 setfillstyle(SOLID_FILL,CYAN);
 rectangle(75,150,125,250);
 line(100,100,300,75);
 moveto(300,75);
 linerel(25,50);
 line(125,150,325,125);
 line(325,125,325,225);
 line(125,250,325,225);
 rectangle(85,200,115,250);
 circle(400,25,20);
 line(175,175,275,160);
 line(175,175,175,215);
 moveto(275,160);
 linerel(0,40);
 line(175,215,275,200);
 line(175,205,275,190);
 line(175,195,275,180);
 line(175,185,275,170);
 line(85,200,95,210);
 moveto(95,210);
 linerel(0,30);
 line(85,250,95,240);
 outtextxy(120,120,"Shubham's Hut");
 outtextxy(150,110,"Shubham's Hut");
 outtextxy(180,100,"Shubham's Hut");
 circle(100,130,8);
 getch(); }
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

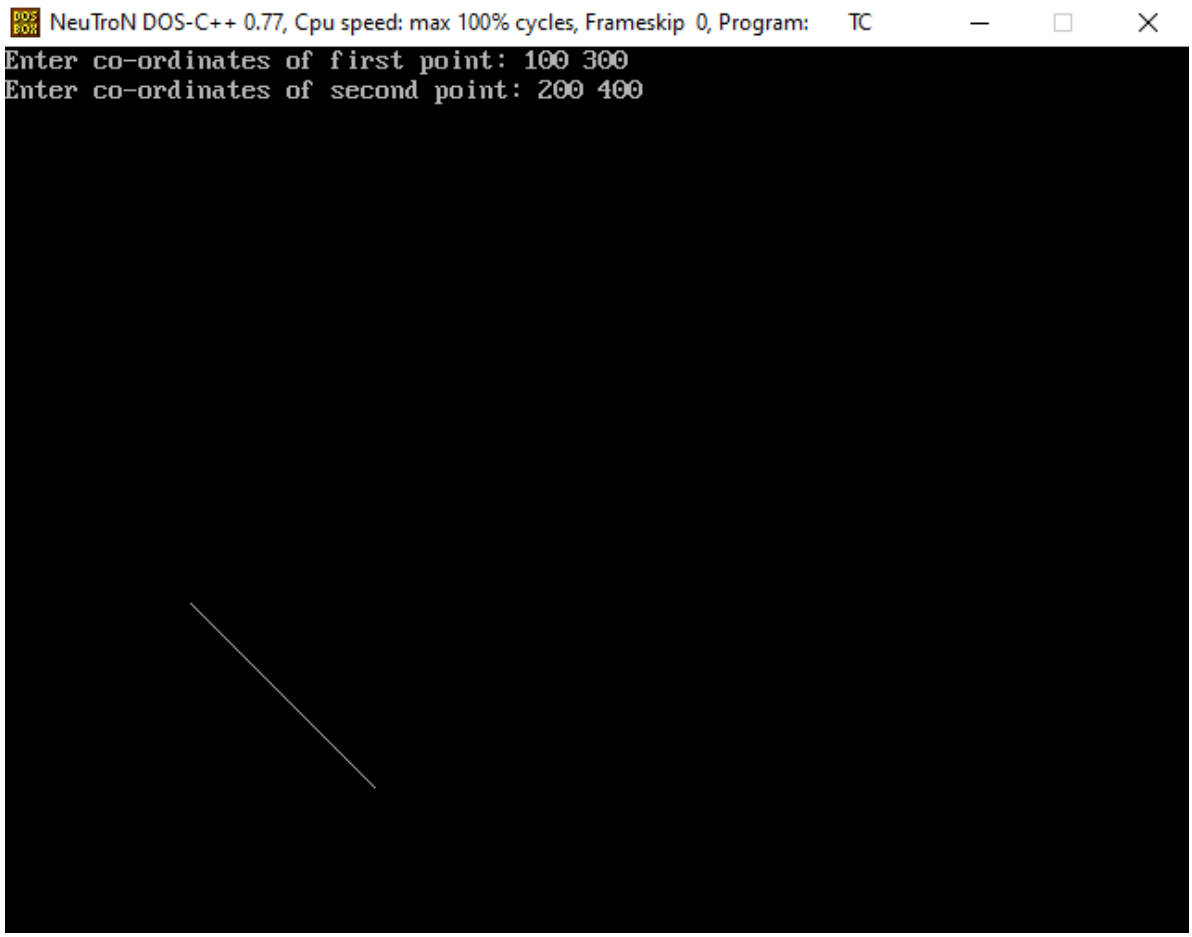
Q 2 - WAP To Make A Line By Using DDA Line Algorithm (m<1 and m>1).

```
#include <graphics.h>
#include <iostream.h>
#include <math.h>
#include <dos.h>
int main( ){
float x,y,x1,y1,x2,y2,dx,dy,step;
int i,gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
cout<<"Enter the value of x1 and y1 : ";
cin>>x1>>y1;
cout<<"Enter 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,5);
x=x+dx;
y=y+dy;
i=i+1;
delay(100);
}
closegraph();
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the value of x1 and y1 : 100 300
Enter the value of x2 and y2: 300 200

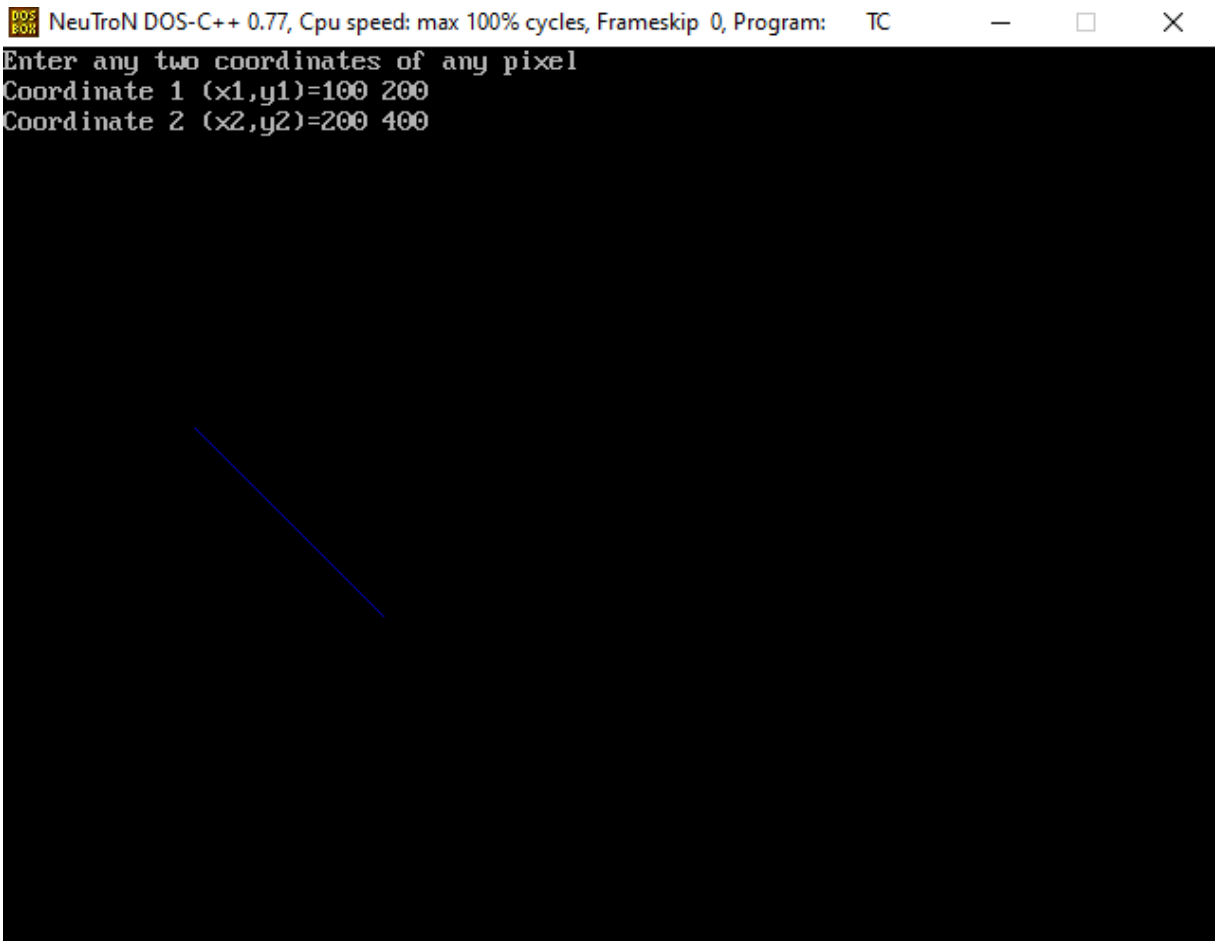
Q 3 - WAP To Make A Line By Using Bresenhams Line Algorithm .

```
#include<iostream.h>
#include<graphics.h>
void drawline(int x0, int y0, int x1, int y1){
int dx, dy, p, x, y;
dx=x1-x0;
dy=y1-y0;
x=x0;
y=y0;
p=2*dy-dx;
while(x<x1){
if(p>=0){

putpixel(x,y,7);
y=y+1;
p=p+2*dy-2*dx;
}
else{
putpixel(x,y,7);
p=p+2*dy;
}
x=x+1;
}
}
int main(){
int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
cout<<"Enter co-ordinates of first point: ";
cin>>x0>>y0;
cout<<"Enter co-ordinates of second point: ";
cin>>x1>>y1;
drawline(x0, y0, x1, y1);
return 0;
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter co-ordinates of first point: 100 300
Enter co-ordinates of second point: 200 400

Q 4 - WAP To Make A Line By Using Mid-Point Line Algorithm .

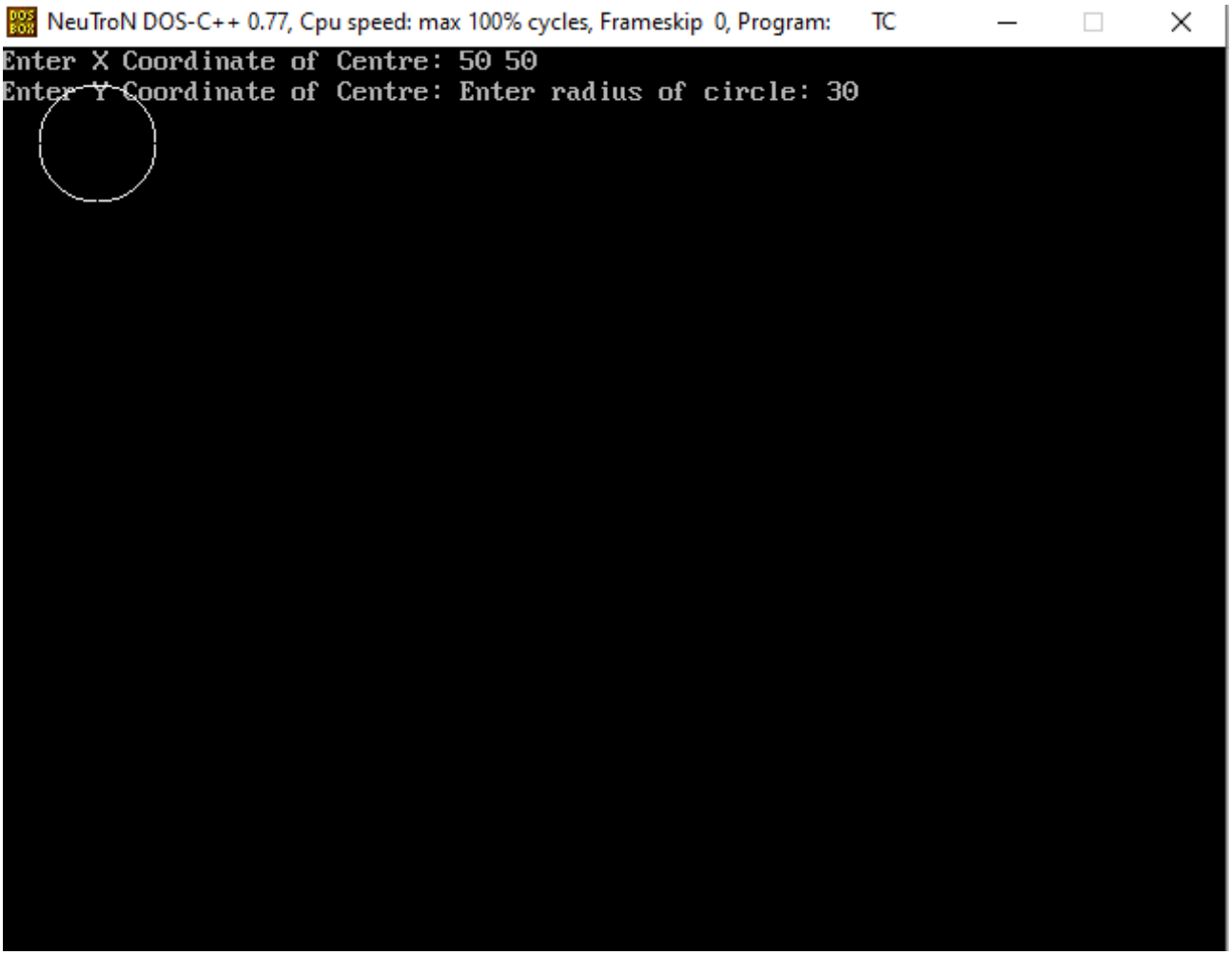
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void midPoint(int X1, int Y1, int X2, int Y2)
{
int dx = X2 - X1;
int dy = Y2 - Y1;
int d = dy - (dx/2);
int x = X1, y = Y1;
putpixel(x,y);
cout << x << "," << y << "\n";
while (x < X2)
{ x++;
if (d < 0)
d = d + dy;
else
{
d += (dy - dx);
y++;
}
putpixel(x,y);
}
}
int main()
{
int gd = DETECT, gm;
initgraph(&gm,&gd,"C:\\TC\\BGI");
int X1, Y1, X2, Y2;
printf("Enter co-ordinates of first point: ");
scanf("%d %d",&X1, &Y1);
printf("Enter co-ordinates of second point: ");
scanf("%d %d",&X2, &Y2);
midPoint(X1, Y1, X2, Y2);
getch();
closegraph();
return 0;}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter any two coordinates of any pixel
Coordinate 1 (x1,y1)=100 200
Coordinate 2 (x2,y2)=200 400

Q 5 - WAP To Make A Circle by Using Bresenham's Circle Algorithm.

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main(){
int gd=DETECT,gm;
int x,y,a,b,radius,d,flag;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI\\");
printf("Enter X Coordinate of Centre: ");
scanf("%d",&a);
printf("Enter Y Coordinate of Centre: ");
scanf("%d",&b);
printf("Enter radius of circle: ");
scanf("%d",&radius);
x = 0;
y = radius;
d = 3 - (2*r);
while(x<=y){
putpixel(a+x,b-y,15);
if(d<0){
d = d+(4*x)+6;
}
else{
d = d+(4*(x-y))+10;
y--;
}
x++;
putpixel(a-x,b+y,15);
putpixel(a+x,b+y,15);
putpixel(a-x,b-y,15);
putpixel(a+y,b-x,15);
putpixel(a-y,b+x,15);
putpixel(a+y,b+x,15);
putpixel(a-y,b-x,15);
}
getch();
```

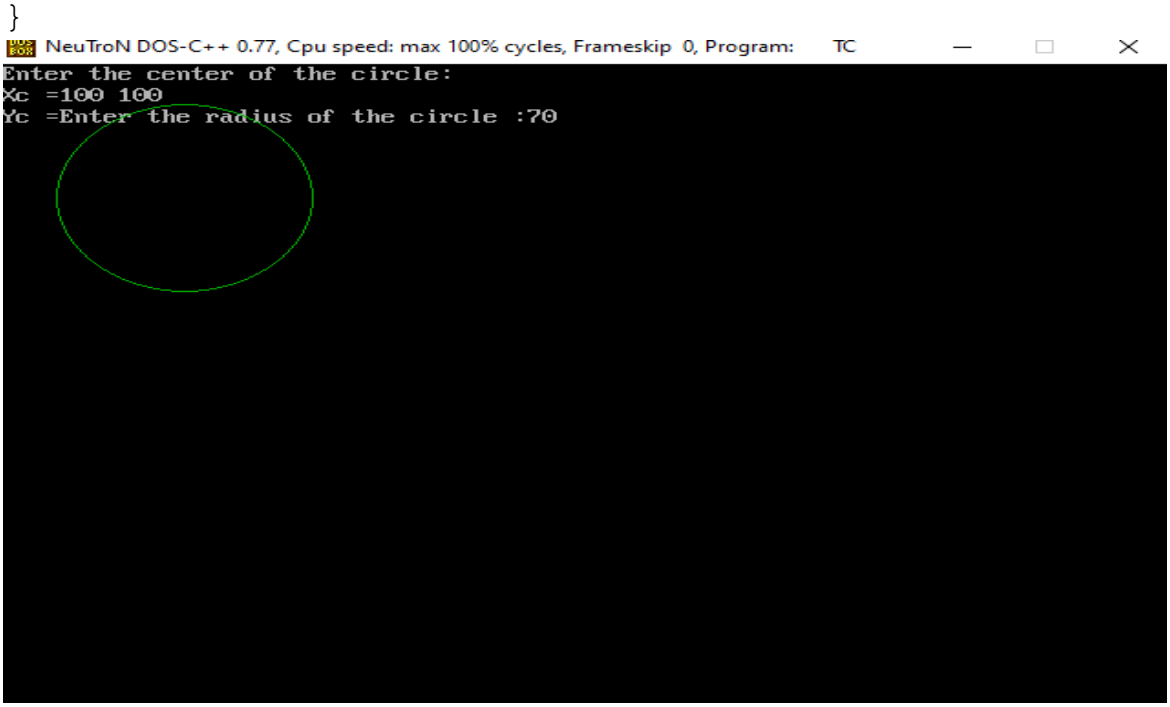
```
closegraph();
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter X Coordinate of Centre: 50 50
Enter Y Coordinate of Centre: Enter radius of circle: 30

Q 6 - WAP To Make A Circle by Using Mid-Point Circle Algorithm.

```
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void draw_circle(int,int,int);
void symmetry(int,int,int,int);
void main()
{ int xc,yc,R;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("Enter the center of the circle:\n");
printf("Xc =");
scanf("%d",&xc);
printf("Yc =");
scanf("%d",&yc);
printf("Enter the radius of the circle :");
scanf("%d",&R);
draw_circle(xc,yc,R);
getch();
closegraph();
}
void draw_circle(int xc,int yc,int rad)
{
int x = 0;
int y = rad;
int p = 1-rad;
symmetry(x,y,xc,yc);
for(x=0;y>x;x++)
{
if(p<0)
p += 2*x + 3;
else
{ p+= 2*(x-y) + 5;

y--;
}
```

```
symmetry(x,y,xc,yc);
delay(50);
}
}
void symmetry(int x,int y,int xc,int yc)
{
putpixel(xc+x,yc-y,GREEN); //For pixel (x,y)
delay(50);
putpixel(xc+y,yc-x, GREEN); //For pixel (y,x)
delay(50);
putpixel(xc+y,yc+x, GREEN); //For pixel (y,-x)
delay(50);
putpixel(xc+x,yc+y, GREEN); //For pixel (x,-y)
delay(50);
putpixel(xc-x,yc+y, GREEN); //For pixel (-x,-y)
delay(50);
putpixel(xc-y,yc+x, GREEN); //For pixel (-y,-x)
delay(50);
putpixel(xc-y,yc-x, GREEN); //For pixel (-y,x)
delay(50);
putpixel(xc-x,yc-y, GREEN); //For pixel (-x,y)
delay(50);
}
```



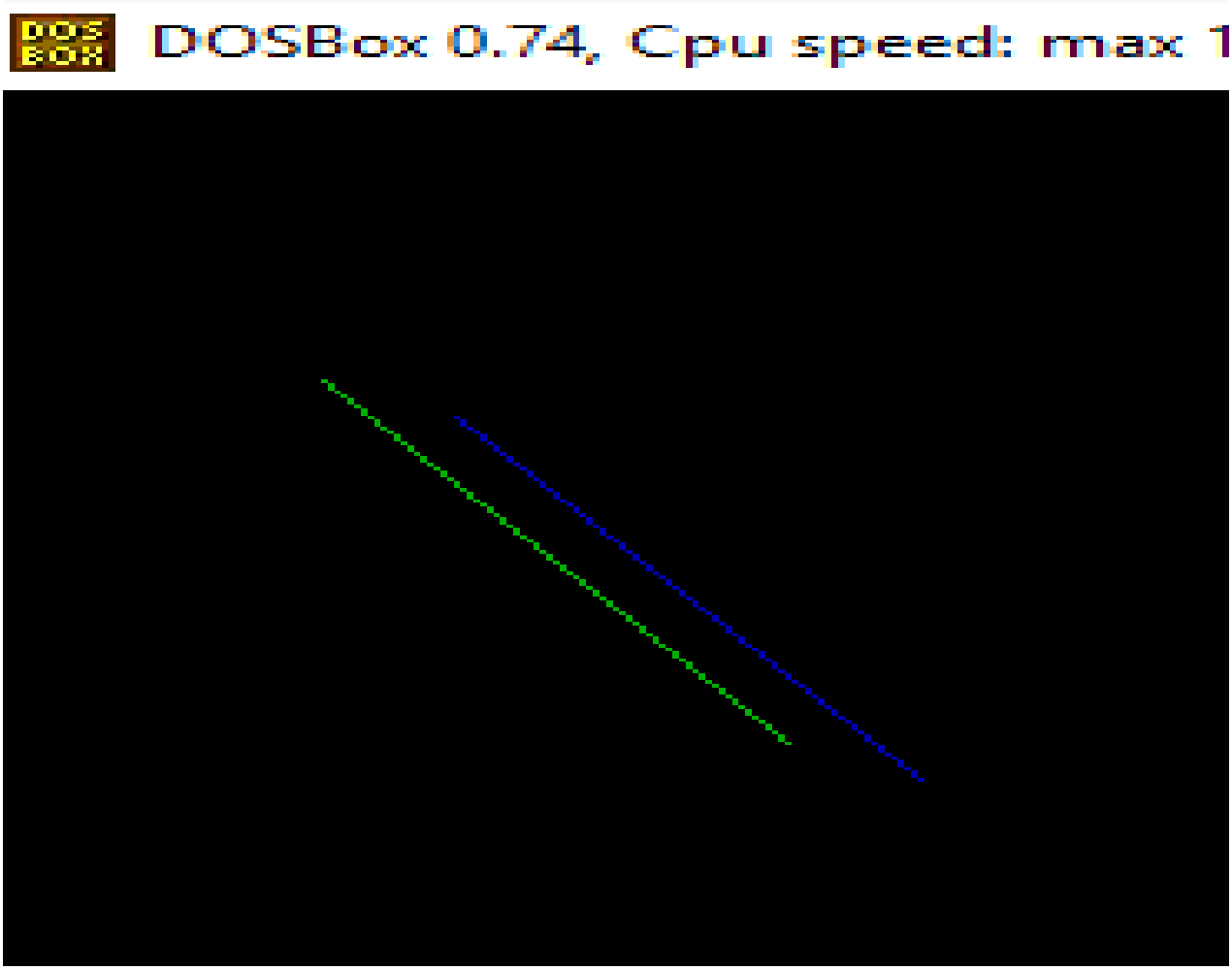
Q 7 - WAP To Make An Ellipse By Using Mid-Point Ellipse Algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void ellipse(int xc,int yc,int rx,int ry){
int gm=DETECT,gd;
int x, y, p;
clrscr();
initgraph(&gm,&gd,"C:\\TC\\BGI");
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=x+1;
p=p+(2*ry*ry*x)+(ry*ry);
}
else
{
x=x+1;
y=y-1;
p=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=y-1;
p=p-(2*rx*rx*y)+(rx*rx);

}
else
{
y=y-1;
x=x+1;
p=p+(2*ry*ry*x)-(2*rx*rx*y)-(rx*rx);
}
}
getch();
closegraph();
}
void main()
{
int xc,yc,rx,ry;
clrscr();
printf("Enter Xc=");
scanf("%d",&xc);
printf("Enter Yc=");
scanf("%d",&yc);
printf("Enter Rx=");
scanf("%d",&rx);
printf("Enter Ry=");
scanf("%d",&ry);
ellipse(xc,yc,rx,ry);
getch();
}
```

Q.8 – WAP for Translation in 2 – Dimension.

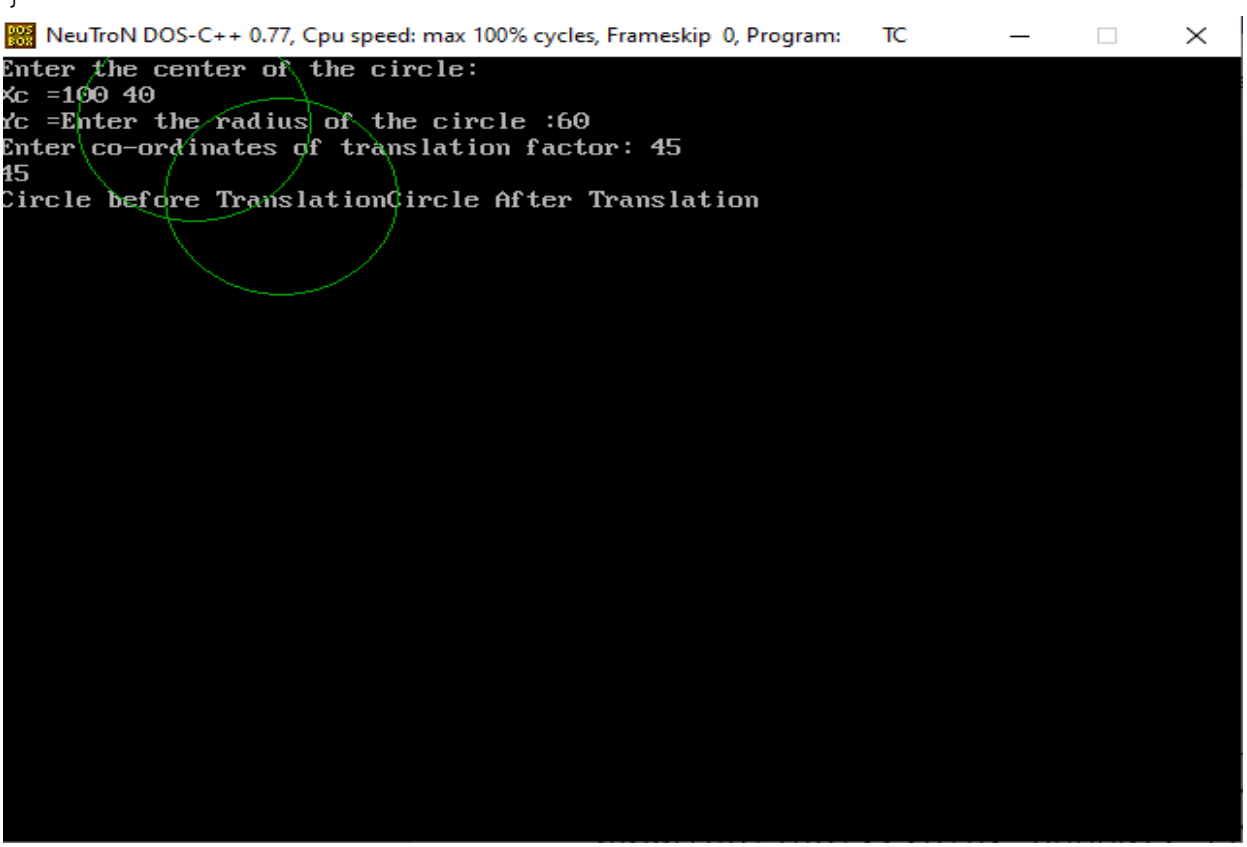
```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>

int main()
{
int gd = DETECT, gm;
initgraph(&gm,&gd,"C:\\TC\\BGI");
int X1, Y1, X2, Y2;
printf("Enter co-ordinates of first point: ");
scanf("%d %d",&X1, &Y1);
printf("Enter co-ordinates of second point: ");
scanf("%d %d",&X2, &Y2);
printf("Enter co-ordinates of translation factor: ");
scanf("%d %d",&Tx, &Ty);

printf("\nLine before Translation”);

line(X1,Y1,X2,Y2);

X1+=Tx;
X2+=Tx;
Y1+=Ty;
Y2+=Ty;

printf("\nLine after Translation”);

line(X1,Y1,X2,Y2);

getch();

closegraph();

}
```


DOSBox 0.74, Cpu speed: max 1

Q.9 – WAP for Translation in 2 – Dimension. (Circle)

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void draw_circle(int,int,int);
void symmetry(int,int,int,int);
void main()
{ int xc,yc,R;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("Enter the center of the circle:\n");
printf("Xc =");
scanf("%d",&xc);
printf("Yc =");
scanf("%d",&yc);
printf("Enter the radius of the circle :");
scanf("%d",&R);
printf("Enter co-ordinates of translation factor: ");
scanf("%d %d",&Tx, &Ty);
printf("Circle before Translation");
draw_circle(xc,yc,R);
xc+=Tx;
yc+=Ty;
printf("Circle After Translation");
draw_circle(xc,yc,R);

getch();
closegraph();
}
void draw_circle(int xc,int yc,int rad)
{
int x = 0;
int y = rad;
int p = 1-rad;
symmetry(x,y,xc,yc);
for(x=0;y>x;x++)
{
if(p<0)
```

```
p += 2*x + 3;
else
     {
     p+= 2*(x-y) + 5;
     y--;
}
symmetry(x,y,xc,yc);
}
}
void symmetry(int x,int y,int xc,int yc)
{
putpixel(xc+x,yc-y,GREEN); //For pixel (x,y)
putpixel(xc+y,yc-x, GREEN); //For pixel (y,x)
putpixel(xc+y,yc+x, GREEN); //For pixel (y,-x)
putpixel(xc+x,yc+y, GREEN); //For pixel (x,-y)
putpixel(xc-x,yc+y, GREEN); //For pixel (-x,-y)
putpixel(xc-y,yc+x, GREEN); //For pixel (-y,-x)
putpixel(xc-y,yc-x, GREEN); //For pixel (-y,x)
putpixel(xc-x,yc-y, GREEN); //For pixel (-x,y)
}
```



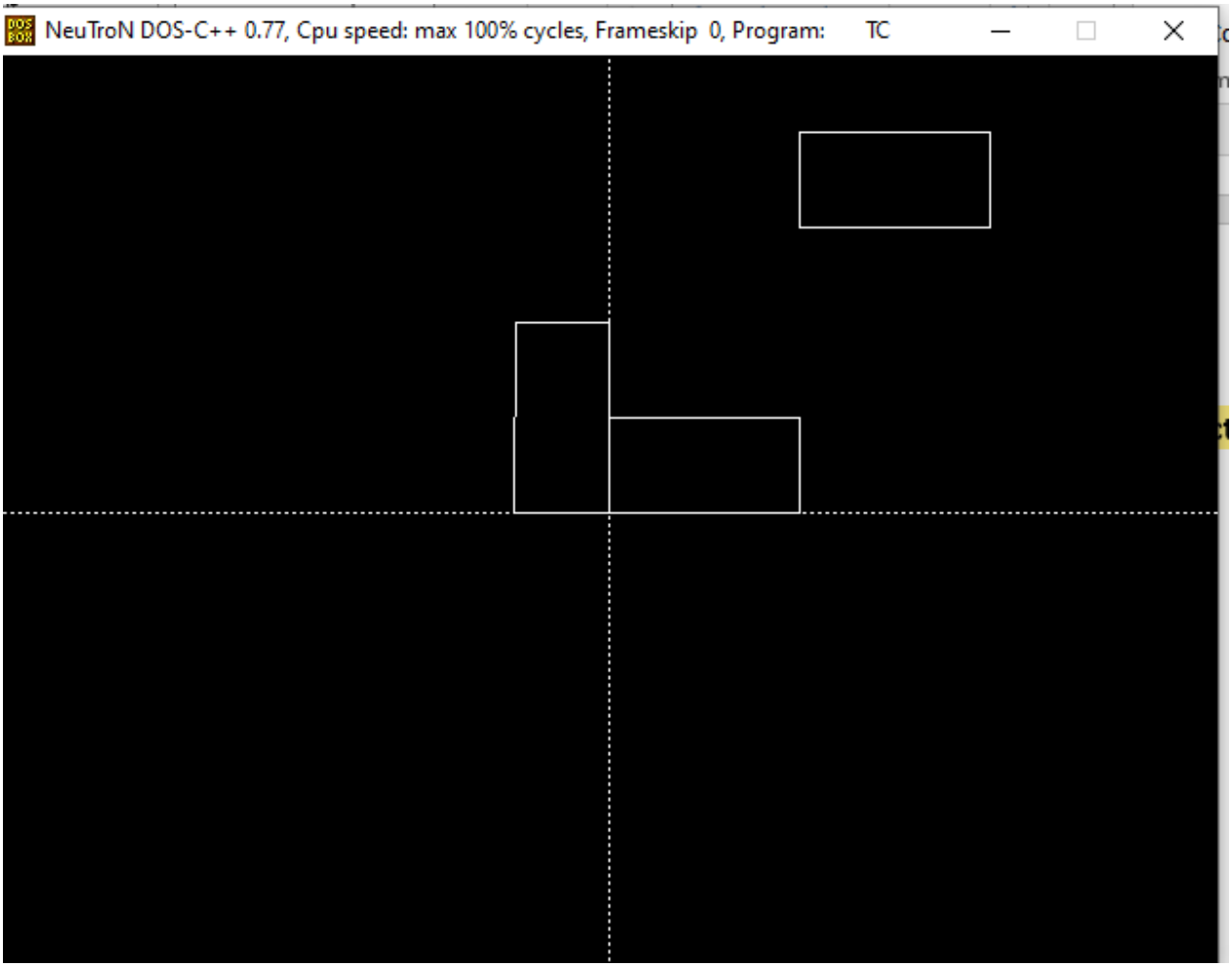
Q 10 - WAP To Rotate A Rectangle At One Of Its Coordinate In Clockwise Direction.

```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
#include<math.h>
#include<iostream.h>
#include<conio.h>
void draw (int r[][2])
{ int i;
setlinestyle (DOTTED_LINE, 0, 1);
line (320, 0, 320, 480);
line (0, 240, 640, 240);
setlinestyle (SOLID_LINE, 0, 1);
line (320+r[0][0], 240-r[0][1], 320+r[1][0], 240-
r[1][1]);
line (320+r[0][0], 240-r[0][1], 320+r[3][0], 240-
r[3][1]);
line (320+r[1][0], 240-r[1][1], 320+r[2][0], 240-
r[2][1]);
line (320+r[2][0], 240-r[2][1], 320+r[3][0], 240-
r[3][1]);
}
void reset (int r[][2])
{
int i;
int val[4][2] = {

{ 0, 0 },{ 100, 0 },{ 100, 50 },{ 0, 50 }
};
for (i=0; i<4; i++)
{ r[i][0] = val[i][0];
r[i][1] = val[i][1];
}
}
void rotate (int r[][2], int angle)
{
```

```
int i;
double ang_rad = (angle * M_PI) / 180;
for (i=0; i<4; i++)
{ double xnew, ynew;
xnew = r[i][0] * cos (ang_rad) - r[i][1] * sin
(ang_rad);
ynew = r[i][0] * sin (ang_rad) + r[i][1] * cos
(ang_rad);
r[i][0] = xnew;
r[i][1] = ynew;
}
}

void translate (int r[][2], int dx, int dy)
{
int i;
for (i=0; i<4; i++)
{
r[i][0] += dx;
r[i][1] += dy;
}
}
void int()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"..//bgi");
}
void main()
{ int r[4][2],angle,dx,dy,x, y,choice;
do
{
clrscr();
printf("1.Rotation about an arbitrary point\n");
printf("2.Exit\n\n");
printf("Enter your choice: ");
scanf("%d",&choice);
switch(choice)
{
```

```
case 1: printf("Enter Negative Angle For Clockwise
Rotation ");
scanf("%d",&angle);
printf("Enter the x- and y-coordinates of the point:
");
scanf("%d%d",&x,&y);
ini();
cleardevice();
reset(r);
translate(r,x,y);
draw(r);
putpixel(320+x,240-y,WHITE);
getch();
translate(r,-x,-y);
draw(r);getch();
rotate(r,angle);
draw(r);getch();
translate(r,x,y);
cleardevice();
draw(r);
putpixel(320+x,240-y,WHITE);
getch();
closegraph();

break;
case 2: closegraph();
}
}while(choice!=2);
```

}

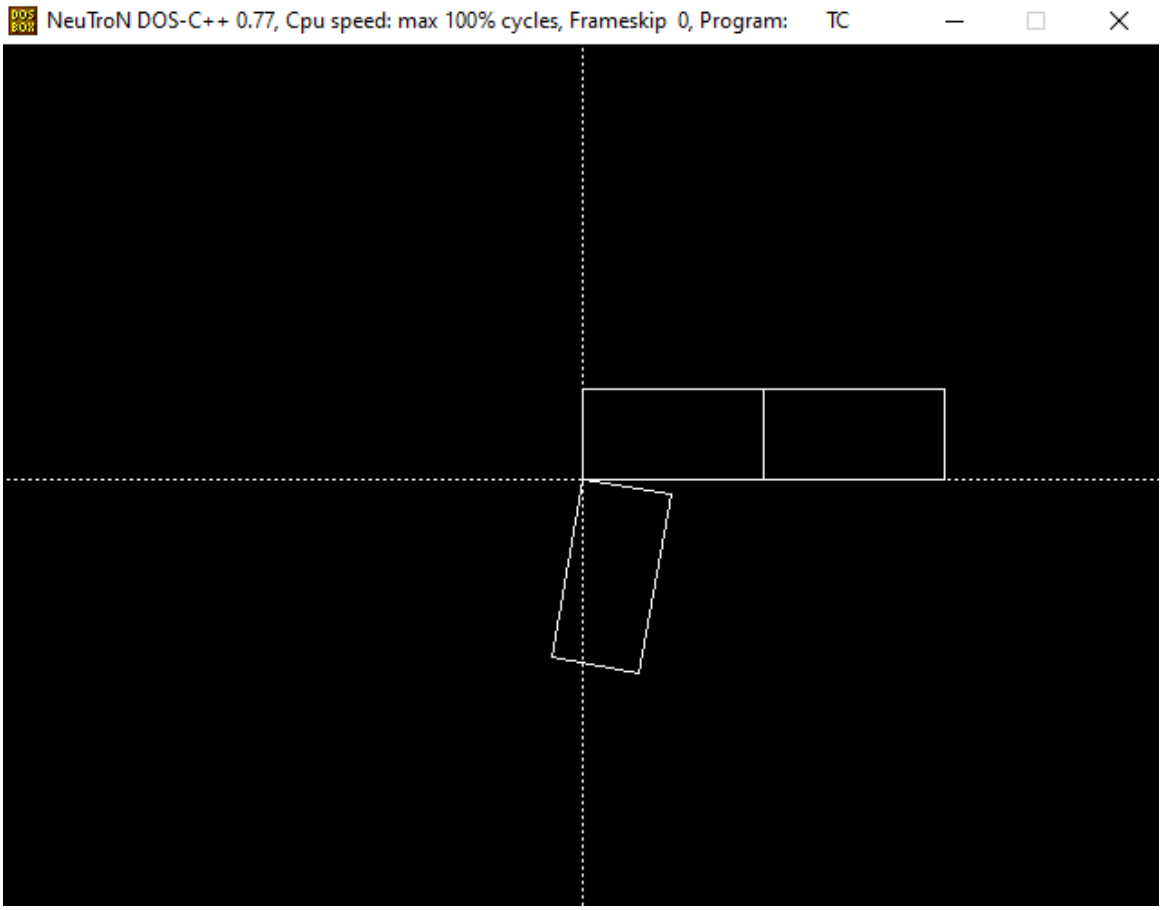

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

Q 11 - WAP To Rotate A Rectangle At One Of Its Coordinate In Anticlockwise Direction.

```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
#include<math.h>
#include<iostream.h>
#include<conio.h>
void draw (int r[][2])
{ int i;
setlinestyle (DOTTED_LINE, 0, 1);
line (320, 0, 320, 480);
line (0, 240, 640, 240);
setlinestyle (SOLID_LINE, 0, 1);
line (320+r[0][0], 240-r[0][1], 320+r[1][0], 240-
r[1][1]);
line (320+r[0][0], 240-r[0][1], 320+r[3][0], 240-
r[3][1]);
line (320+r[1][0], 240-r[1][1], 320+r[2][0], 240-
r[2][1]);
line (320+r[2][0], 240-r[2][1], 320+r[3][0], 240-
r[3][1]);
}
void reset (int r[][2])
{
int i;
int val[4][2] = {

{ 0, 0 },{ 100, 0 },{ 100, 50 },{ 0, 50 }
};
for (i=0; i<4; i++)
{
r[i][0] = val[i][0];
r[i][1] = val[i][1];
}
}
void rotate (int r[][2], int angle)
```

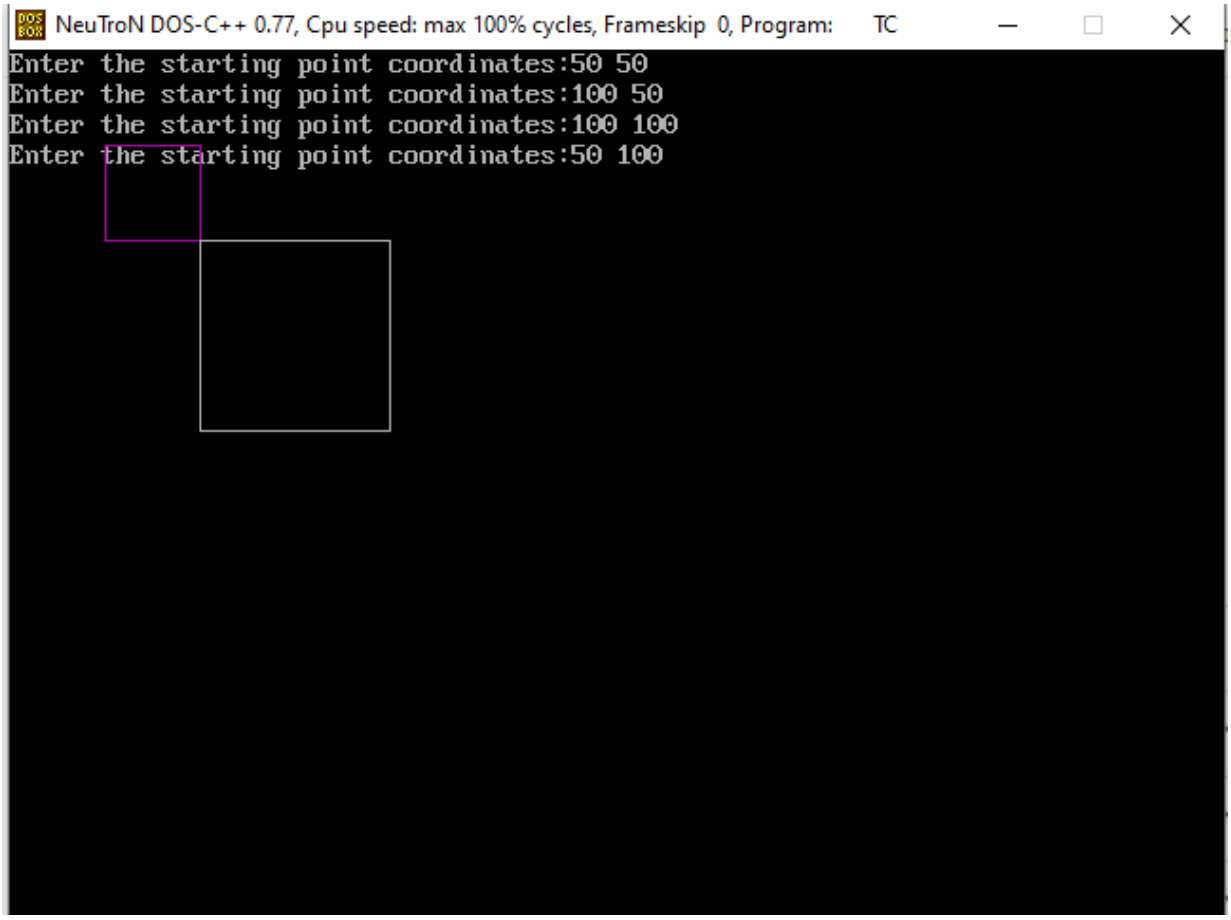
```
{
int i;
double ang_rad = (angle * M_PI) / 180;
for (i=0; i<4; i++)
{
double xnew, ynew;
xnew = r[i][0] * cos (ang_rad) - r[i][1] * sin
(ang_rad);
ynew = r[i][0] * sin (ang_rad) + r[i][1] * cos
(ang_rad);
r[i][0] = xnew;
r[i][1] = ynew;

}
}
void translate (int r[][2], int dx, int dy)
{
int i;
for (i=0; i<4; i++)
{
r[i][0] += dx;
r[i][1] += dy;
}
}
void ini()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"..//bgi");
}
void main()
{
int r[4][2],angle,dx,dy,x, y,choice;
do
{ clrscr();
printf("1.Rotation about an arbitrary point\n");
printf("2.Exit\n\n");
printf("Enter your choice: ");
scanf("%d",&choice);
```

```
switch(choice)
{ case 1: printf("Enter Positive Angle For Anti-
Clockwise Rotation ");

scanf("%d",&angle);
printf("Enter the x- and y-coordinates of the point:
");
scanf("%d%d",&x,&y);
ini();
cleardevice();
reset(r);
translate(r,x,y);
draw(r);
putpixel(320+x,240-y,WHITE);
getch();
translate(r,-x,-y);
draw(r);getch();
rotate(r,angle);
draw(r);getch();
translate(r,x,y);
cleardevice();
draw(r);
putpixel(320+x,240-y,WHITE);

getch();
closegraph();
break;
case 2: closegraph();
}
}while(choice!=2);
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

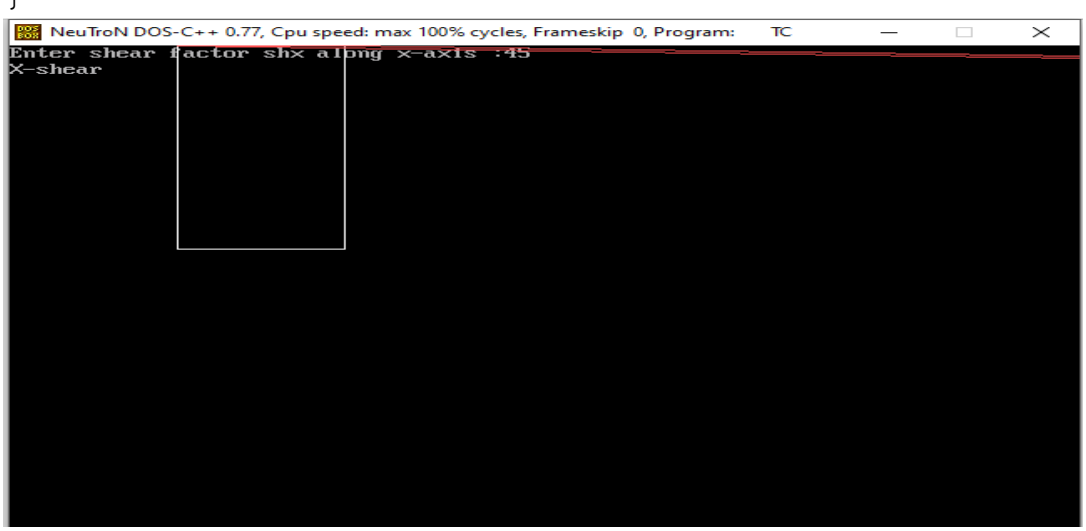
Q 12 - WAP To Scale A Square To Double Its Size.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int gd=DETECT,gm;
float x[4],y[4],sx=2,sy=2;
int i;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
for(i=0;i<4;i++)
{
printf("Enter the starting point coordinates:");
scanf("%f %f",&x[i],&y[i]);
}
setcolor(5);
line(x[0],y[0],x[1],y[1]);
line(x[1],y[1],x[2],y[2]);
line(x[2],y[2],x[3],y[3]);
line(x[3],y[3],x[0],y[0]);

for(i=0;i<4;i++)
{
x[i]=x[i]*sx;
y[i]=y[i]*sy;
}
setcolor(7);

line(x[0],y[0],x[1],y[1]);
line(x[1],y[1],x[2],y[2]);
line(x[2],y[2],x[3],y[3]);
line(x[3],y[3],x[0],y[0]);
getch();
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter the starting point coordinates:50 50
Enter the starting point coordinates:100 50
Enter the starting point coordinates:100 100
Enter the starting point coordinates:50 100

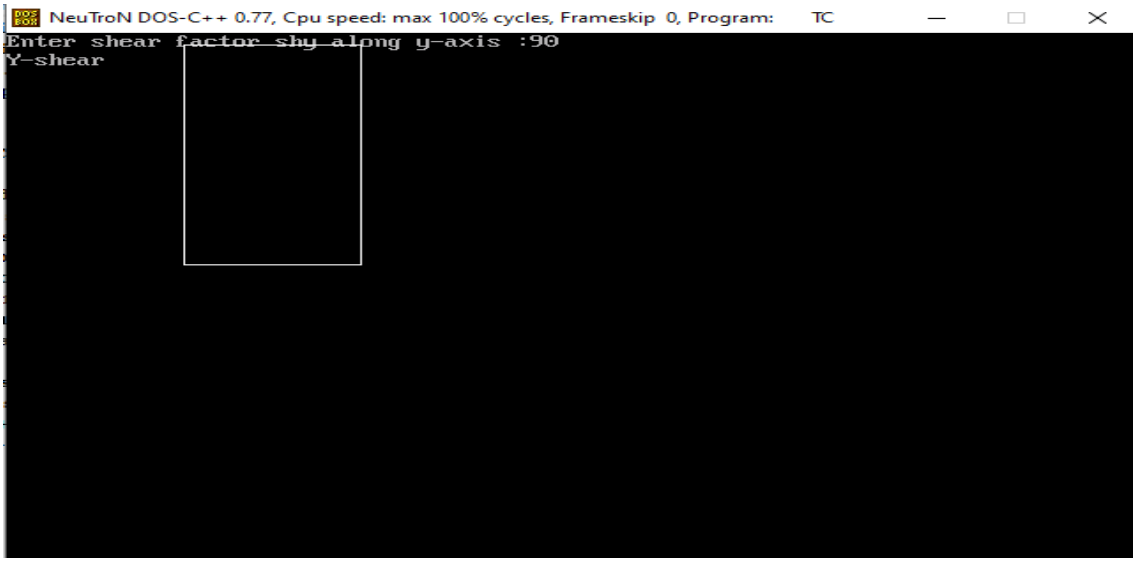
Q 14 - WAP To Shear A Square In X-Direction.

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
float shx,shy;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("Enter shear factor shx along x-axis :");
scanf("%f",&shx);
line(100,0,200,0);
line(200,0,200,200);
line(200,200,100,200);

line(100,200,100,0);
printf("X-shear");
setcolor(12);
line((100+(0*shx)),0,(200+(0*shx)),0);
line((200+(0*shx)),0,(200+(200*shx)),200);
line((200+(200*shx)),200,(100+(200*shx)),200);

line((100+(200*shx)),200,(100+(0*shx)),0);
getch();
}
```



Q 15 - WAP to shear a square in Y-direction .

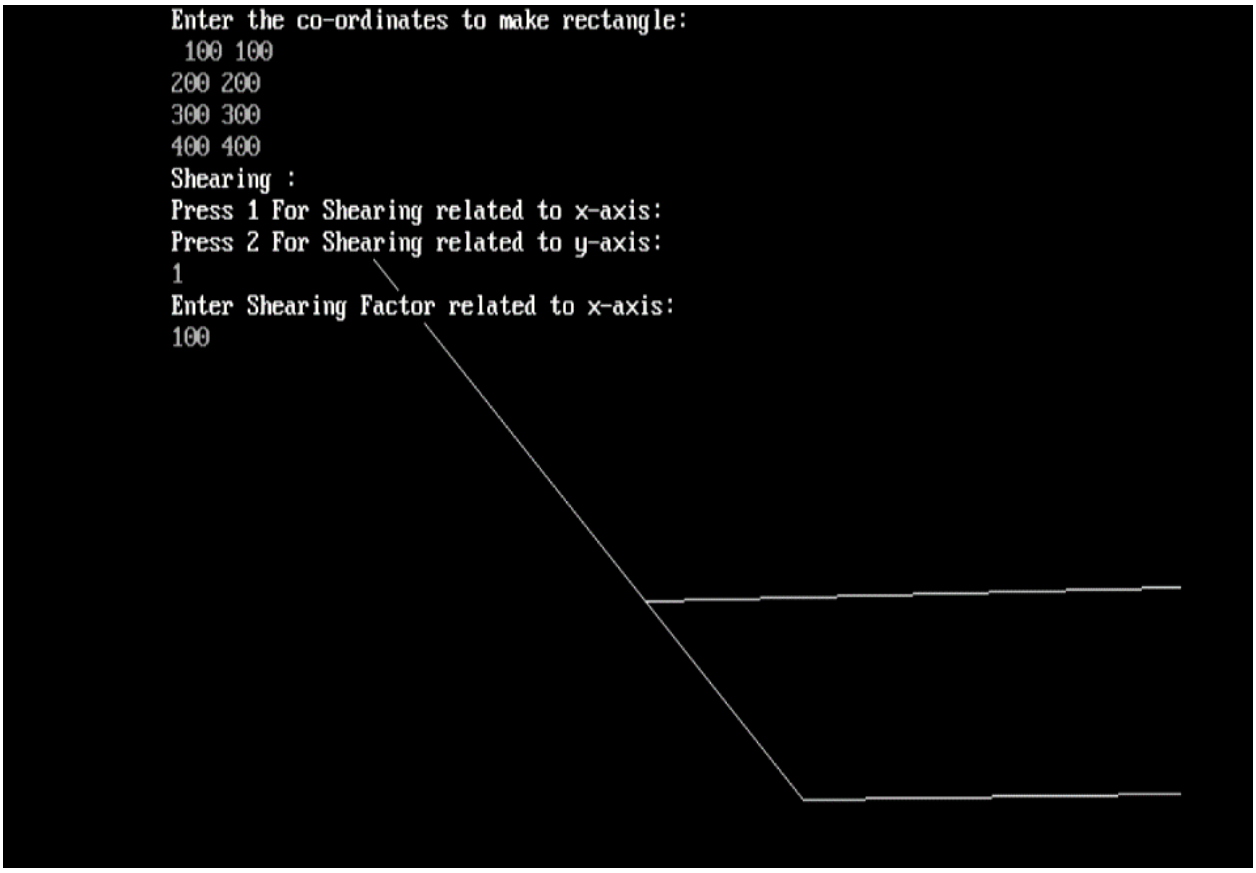
```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
float shx,shy;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("Enter shear factor shy along y-axis :");
scanf("%f",&shy);
line(100,10,200,10);
line(200,10,200,200);
line(200,200,100,200);
line(100,200,100,10);
printf("Y-shear");
setcolor(12);
line(100,10+(shy*100),200,10+(shy*200));
line(200,10+(shy*200),200,200+(shy*200));
line(200,200+(shy*200),100,200+(shy*100));
line(100,200+(shy*100),100,10+(shy*100));
getch();
closegraph();
}
```



Q 16 - WAP To Shear A Square In X And Y Direction.

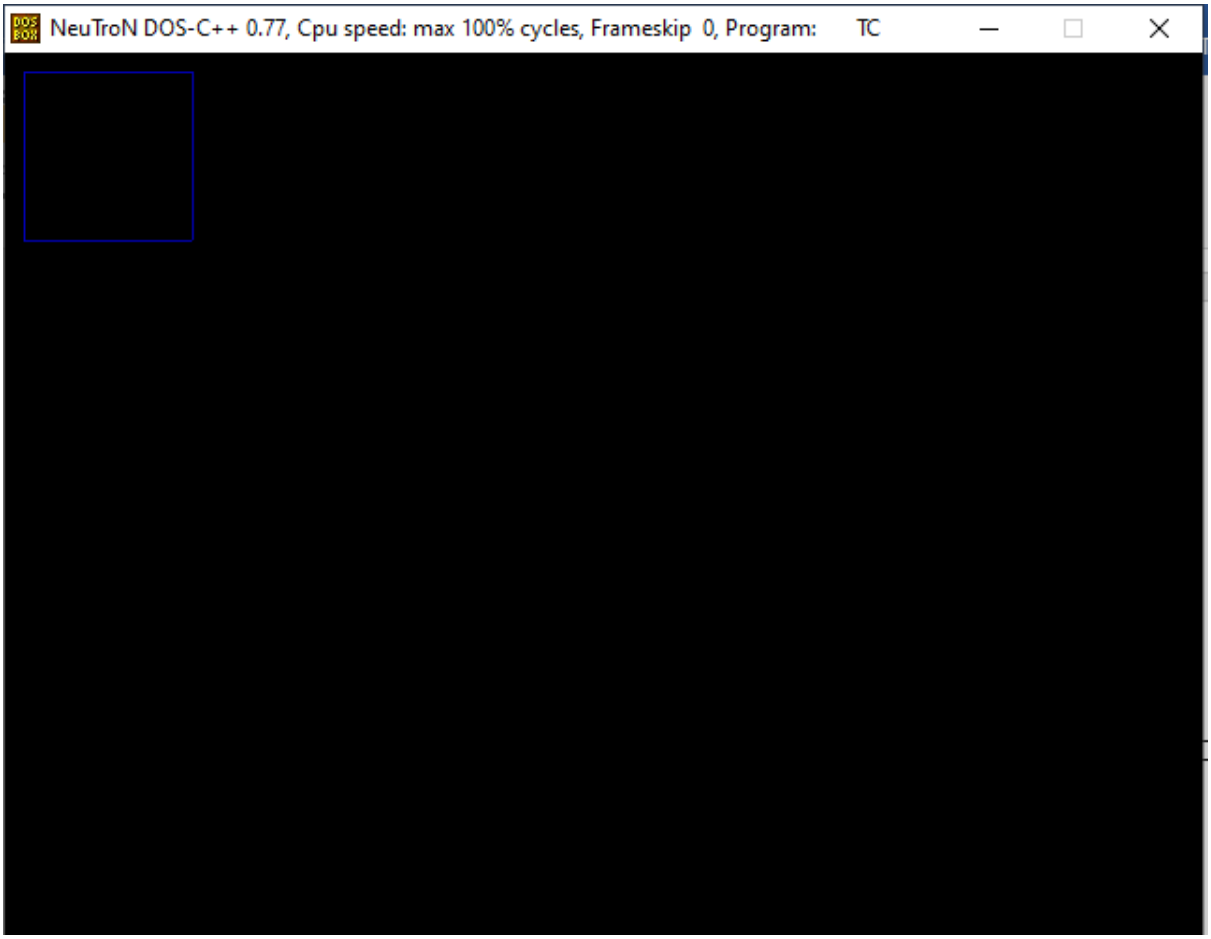
```
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<stdio.h>
void main()
{
int
X,Y,X3,Y3,gd=DETECT,gm,i,shx,shy,x,y,y3,x3,x1,y1,x2,y2,
X1,Y1,X2,Y2;
initgraph(&gd,&gm,"C://turboc3//bgi");
printf("Enter the co-ordinates to make rectangle:\n");
scanf("%d%d%d%d%d%d%d%d",&x,&y,&x1,&y1,&x2,&y2,&x3,&y3)
;
line(x,y,x1,y1);
line(x,y,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
printf("Press 1 For Shearing related to x-axis:\n");
printf("Press 2 For Shearing related to y-axis:\n");
scanf("%d",&i);
switch(i)
{ case 1:
{ printf("Enter Shearing Factor related to x-axis:\n");
scanf("%d",&shx);
X=x+(shx*y);
Y=y;

X1=x1+(shx*y1);Y1=y1;

line(X,Y,X1,Y1);
line(X,Y,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,X1,Y1);
break; }
case 2:
{ printf("Enter Shearing Factor related to y-axis:\n");
```

```
scanf("%d",&shy);
X2=x2;Y2=y2+(x2*shy);
X3=x3;Y3=y3+(x3*shy);
line(x,y,x1,y1);
line(x,y,X2,Y2);
line(X2,Y2,X3,Y3);
line(X3,Y3,x1,y1);
break;
}
} getch();
closegraph();

}
```



Q 17 - WAP To Make A Rectangle By Using DDA Line Algorithm .

```
#include<iostream.h>
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>
void dda ( int x1, int y1, int x2, int y2)
{
int dx=abs(x2-x1);int dy=abs(y2-y1);int step;
if(dx>=dy)
step=dx;
else
step=dy;
if(dx > 0)
dx=dx/step;
if(dy > 0)
dy=dy/step;
int x=x1;int y=y1;int i=1;
while(i<=step)
{
putpixel(x,y,1);
x=x+dx;y=y+dy;
i=i+1;
}
}
void main()
{
int x,y,x1,y1,x2,y2;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
cleardevice();
dda(10,10,100,10);
dda(100,10,100,100);
dda(10,100,100,100);
dda(10,10,10,100);
getch();
closegraph();}
```

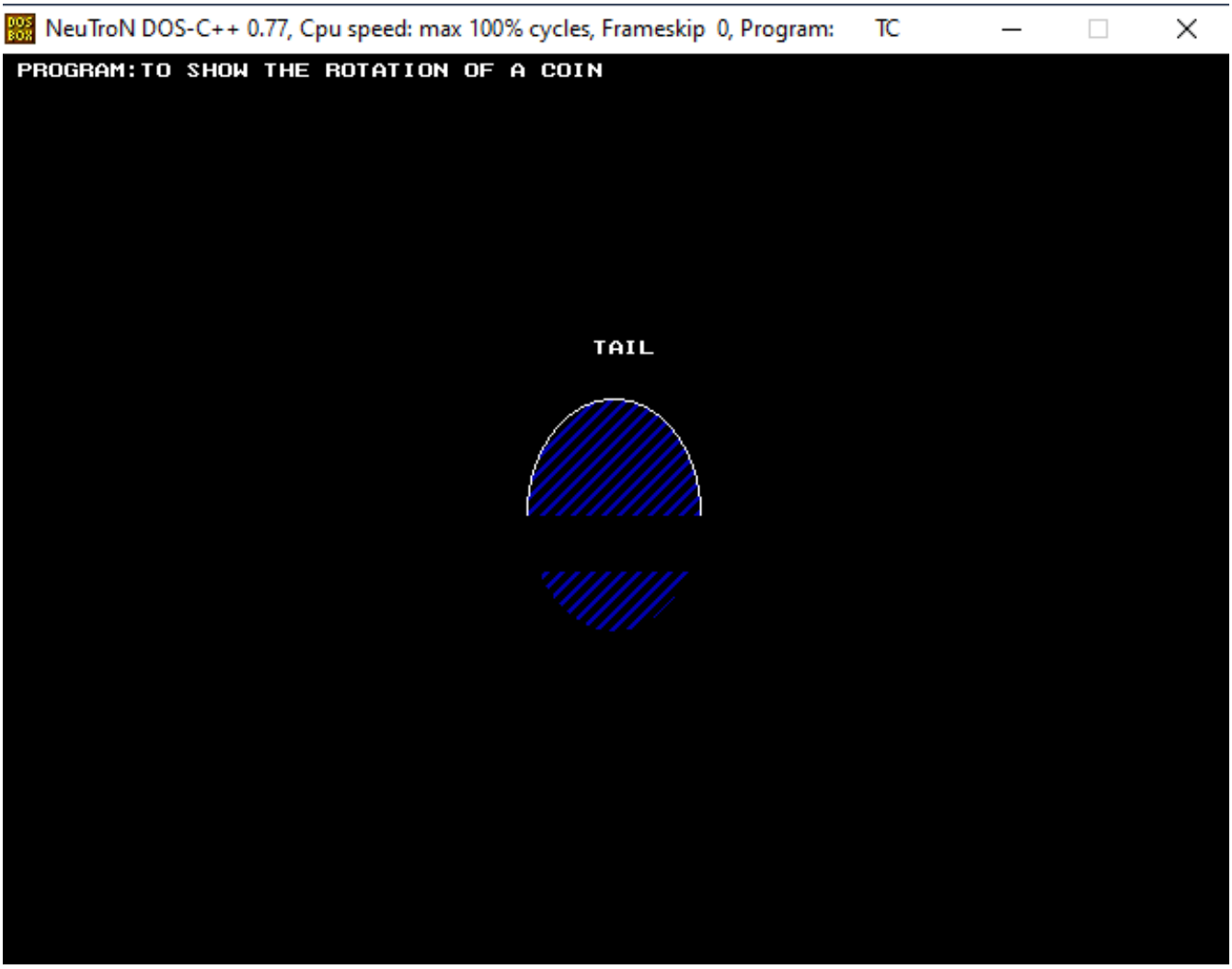

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

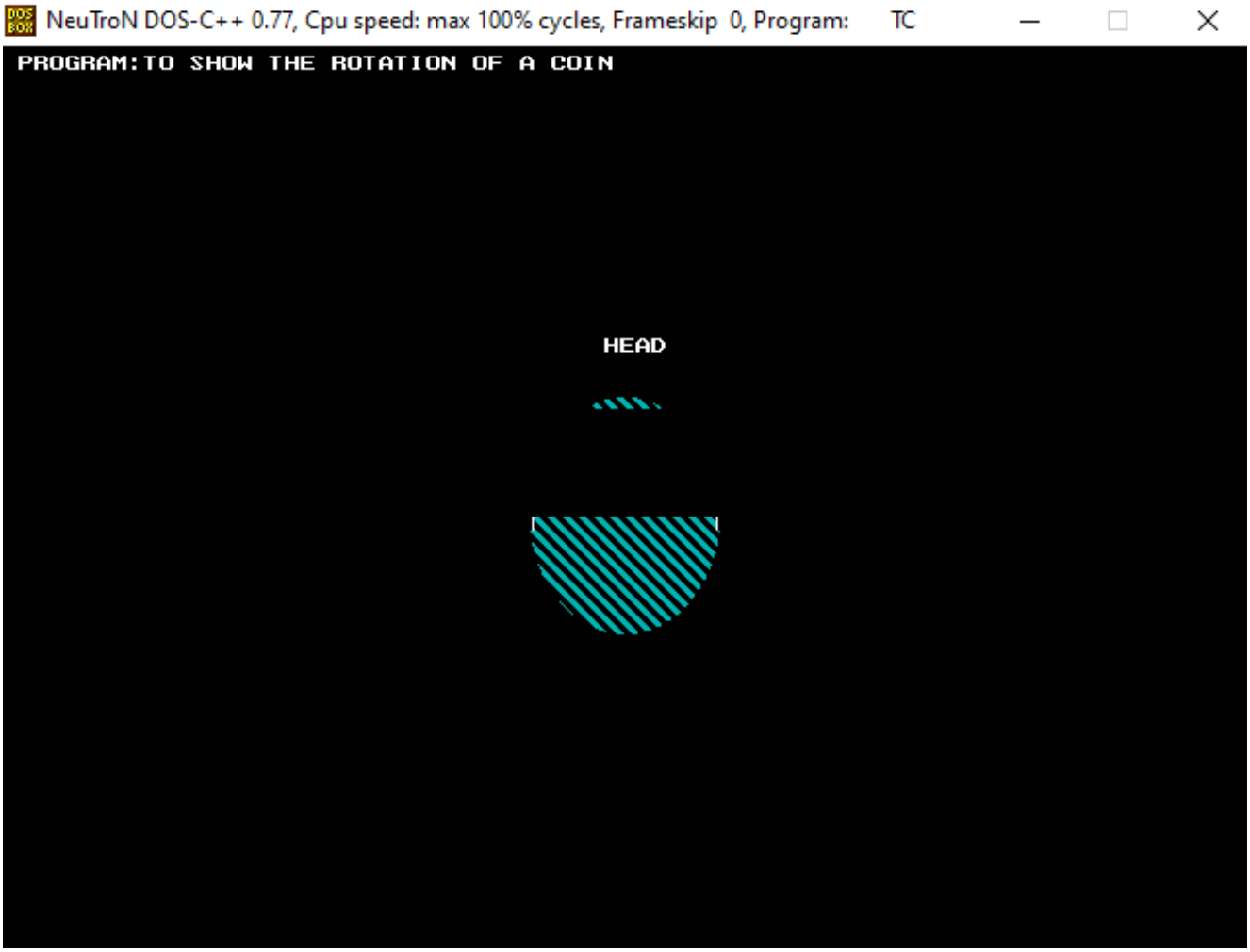
Q 18 - WAP To Rotate A Coin On The Table.

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
#include<graphics.h>
void main(int)
{ int gd=DETECT,gm;
int midx,midy,k=1,b=60,a=60;
int xradius,yradius=60;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
midx = getmaxx() / 2;midy = getmaxy() / 2;
setcolor(getmaxcolor());
while(!kbhit())
{
for(a=60;a>=0;a=a-1)
{
cleardevice();
xradius=a;
if(a==0)
{
k=k+1;
for(b=a;b<=60;b++)
{
cleardevice();
xradius=b;
if(k%2==1)
{
outtextxy(midx-10,midy-90,"TAIL");
setfillstyle(4,1);
}
else
{
outtextxy(midx-10,midy-90,"HEAD");
setfillstyle(5,3);
}
```

```
if(b>0 &&b<57)
{
int xradius1=b-3;
fillellipse(midx,midy,xradius1+1, yradius);
fillellipse(midx,midy,xradius1+2, yradius);
fillellipse(midx,midy,xradius1+3, yradius);

}

outtextxy(10,5, "PROGRAM:TO SHOW THE ROTATION OF A
COIN");
rectangle(230,300,400,320);
rectangle(230,300,250,400);
rectangle(380,300,400,400);
fillellipse(midx,midy,xradius, yradius);
delay(10);
}
}
if(a<57)
{
int xradius1=a+3;
fillellipse(midx,midy,xradius1-1, yradius);
fillellipse(midx,midy,xradius1-2, yradius);
fillellipse(midx,midy,xradius1-3, yradius);
}
if (k%2==1)
{
outtextxy(midx-10,midy-90,"TAIL");
setfillstyle(4,1);
}
else
{
outtextxy(midx-10,midy-90,"HEAD");
setfillstyle(5,3);
}
outtextxy(10,5,"PROGRAM:TO SHOW THE ROTATION OF A
COIN");
fillellipse(midx,midy,xradius, yradius);
```

```
rectangle(230,300,400,320);
rectangle(230,300,250,400);
rectangle(380,300,400,400);
delay(10);
}
}
getch();
closegraph();
```



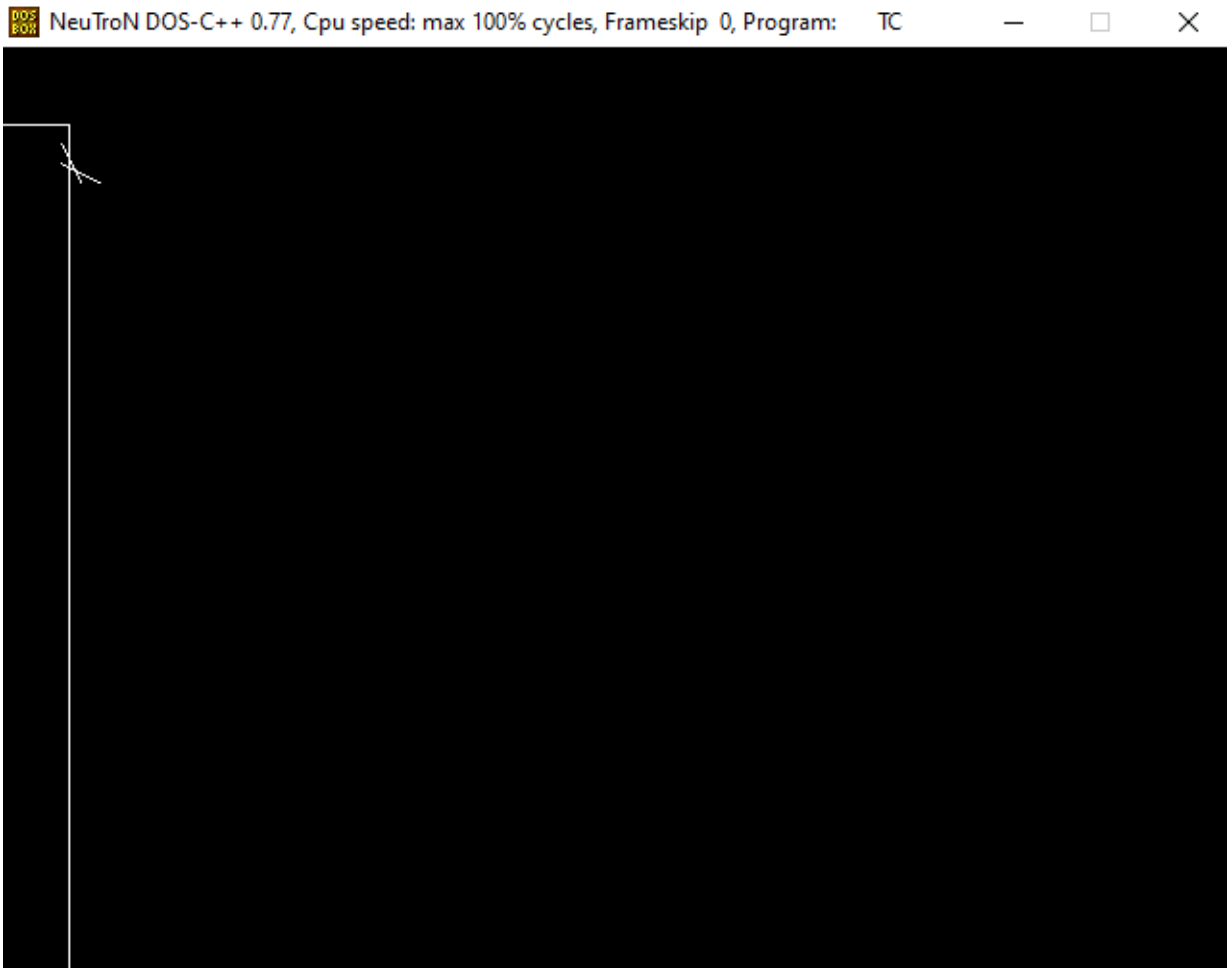

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
PROGRAM:TO SHOW THE ROTATION OF A COIN
HEAD

Q 19 - WAP To Find A Category Of Line By Using Cohen-Sutherland Algorithm.

```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#define MAX 20
enum { TOP = 0x1, BOTTOM = 0x2, RIGHT = 0x4, LEFT = 0x8
};
enum { FALSE, TRUE };
typedef unsigned int outcode;
outcode compute_outcode(int x, int y,
int xmin, int ymin, int xmax, int ymax)
{ outcode oc = 0;
if (y > ymax)
oc |= TOP;
else if (y < ymin)
oc |= BOTTOM;
if (x > xmax)
oc |= RIGHT;
else if (x < xmin)
oc |= LEFT;
return oc;
}
void cohen_sutherland (double x1, double y1, double x2,
double y2,
double xmin, double ymin, double xmax, double ymax)
{ int accept;
int done;
outcode outcode1, outcode2;
accept = FALSE;
done = FALSE;
outcode1 = compute_outcode (x1, y1, xmin, ymin, xmax,
ymax);
outcode2 = compute_outcode (x2, y2, xmin, ymin, xmax,
ymax);
do
{ if (outcode1 == 0 && outcode2 == 0)
{ accept = TRUE;
```

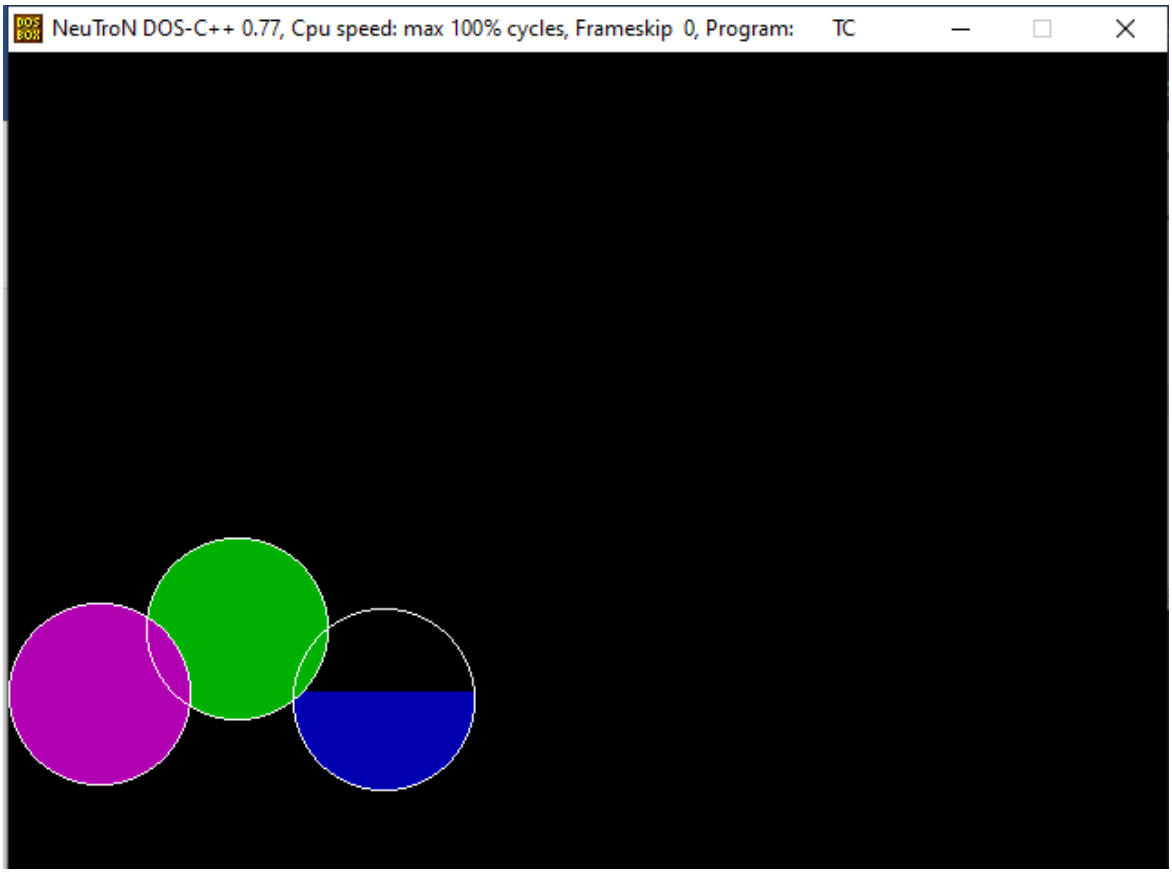
```
done = TRUE;
}

else if (outcode1 & outcode2)
{ done = TRUE;
}
else
{ double x, y;
int outcode_ex = outcode1 ? outcode1 : outcode2;
if (outcode_ex & TOP)
{ x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
y = ymax;
}
else if (outcode_ex & BOTTOM)
{ x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1);
y = ymin;
}
else if (outcode_ex & RIGHT)
{ y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1);
x = xmax;
}
else
{ y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1);
x = xmin;
}
if (outcode_ex == outcode1)
{ x1 = x;y1 = y;

outcode1 = compute_outcode (x1, y1, xmin, ymin,
xmax,ymax);

}
else
{ x2 = x;y2 = y;

outcode2 = compute_outcode (x2, y2, xmin, ymin,
xmax,ymax);
```

```
}
}
} while (done == FALSE);
if (accept == TRUE)
line (x1, y1, x2, y2);
}
void main()
{ int n,i,j;
int ln[MAX][4];int clip[4];
int gd = DETECT, gm;

printf ("Enter the number of lines to be clipped");
scanf ("%d", &n);
printf ("Enter the x- and y-coordinates of the line-
endpoints: \n");
for (i=0; i<n; i++)
for (j=0; j<4; j++)
scanf ("%d", &ln[i][j]);
printf ("Enter the x- and y-coordinates of the left-top
and right-");
printf ("bottom corners\n of the clip window:\n");
for (i=0; i<4; i++)
scanf ("%d", &clip[i]);
initgraph (&gd, &gm, "C:\\TURBOC3\\BGI");
rectangle (clip[0], clip[1], clip[2], clip[3]);
for (i=0; i<n; i++)
line (ln[i][0], ln[i][1], ln[i][2], ln[i][3]);
getch();
cleardevice();
rectangle (clip[0], clip[1], clip[2], clip[3]);
for (i=0; i<n; i++)
{ cohen_sutherland (ln[i][0], ln[i][1], ln[i][2],
ln[i][3],clip[0], clip[1], clip[2], clip[3]);
getch();
} closegraph();
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

Q 20 - WAP To Make Flying Colored Balloons.

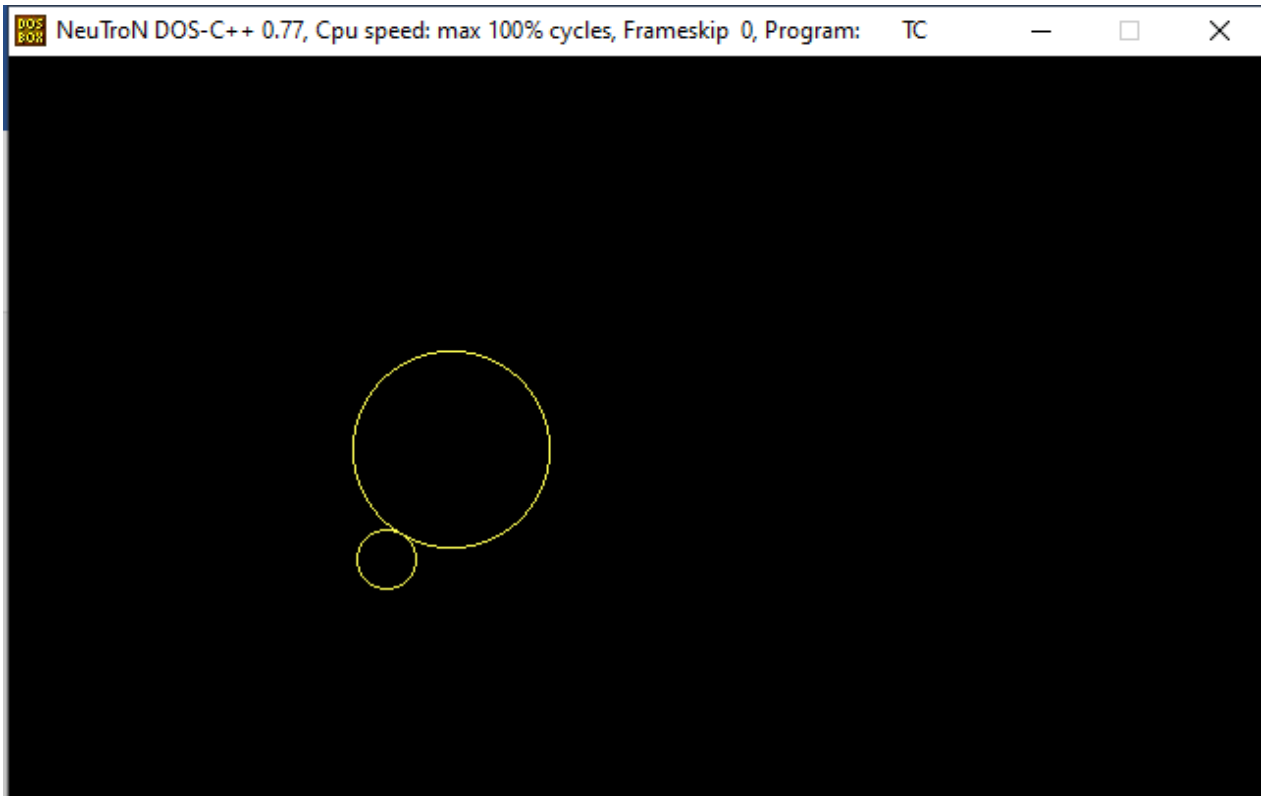
```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
void main()
{
int gd = DETECT, gm;
initgraph(&gd, &gm, "C:\\TURBOC3\\BGI\\");
for(int j=0;j<5;j++)
{
for(int i=0;i<600;i++)
{
setfillstyle(SOLID_FILL,MAGENTA);
circle(50,390-i,50);
floodfill(50,390-i,WHITE);
setfillstyle(SOLID_FILL,GREEN);
circle(90+i,390-2*i,50);
floodfill(90+i,390-2*i,WHITE);
setfillstyle(SOLID_FILL,BLUE);
circle(135+2*i,393-i,50);
floodfill(130+2*i,390-i,WHITE);
setfillstyle(SOLID_FILL,WHITE);
circle(195+2*i,393-3*i,50);
floodfill(195+2*i,393-3*i,WHITE);
delay(5);
cleardevice();
}
}
getch();
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

Q 21 - WAP To Rotate A Circle Outside Another Circle .

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
int xc=225,yc=200,r=50;
int x[3],y[3];
void drawcircles()
{
setcolor(YELLOW);
circle(xc,yc,r);
}
void main()
{
double angle=0,theta;
int i,a;
int gd=DETECT,gm;
initgraph(&gd,&gm,"..\\bgi");
a=0;
while(a<=100)
{
theta=M_PI*angle/180;
cleardevice();
drawcircles();
x[0]=xc+r*cos(theta); //x on circle
y[0]=yc+r*sin(theta); //y on circle
x[1]=xc+(r+15)*cos(theta);// x outside of circle
y[1]=yc+(r+15)*sin(theta);// y outside of circle
x[2]=xc+(r-15)*cos(theta);// x inside circle
y[2]=yc+(r-15)*sin(theta);// y inside circle
angle+=20;
circle(x[1],y[1],15); // for outer circle
a=a+1;
delay(50);
}
getch();
```

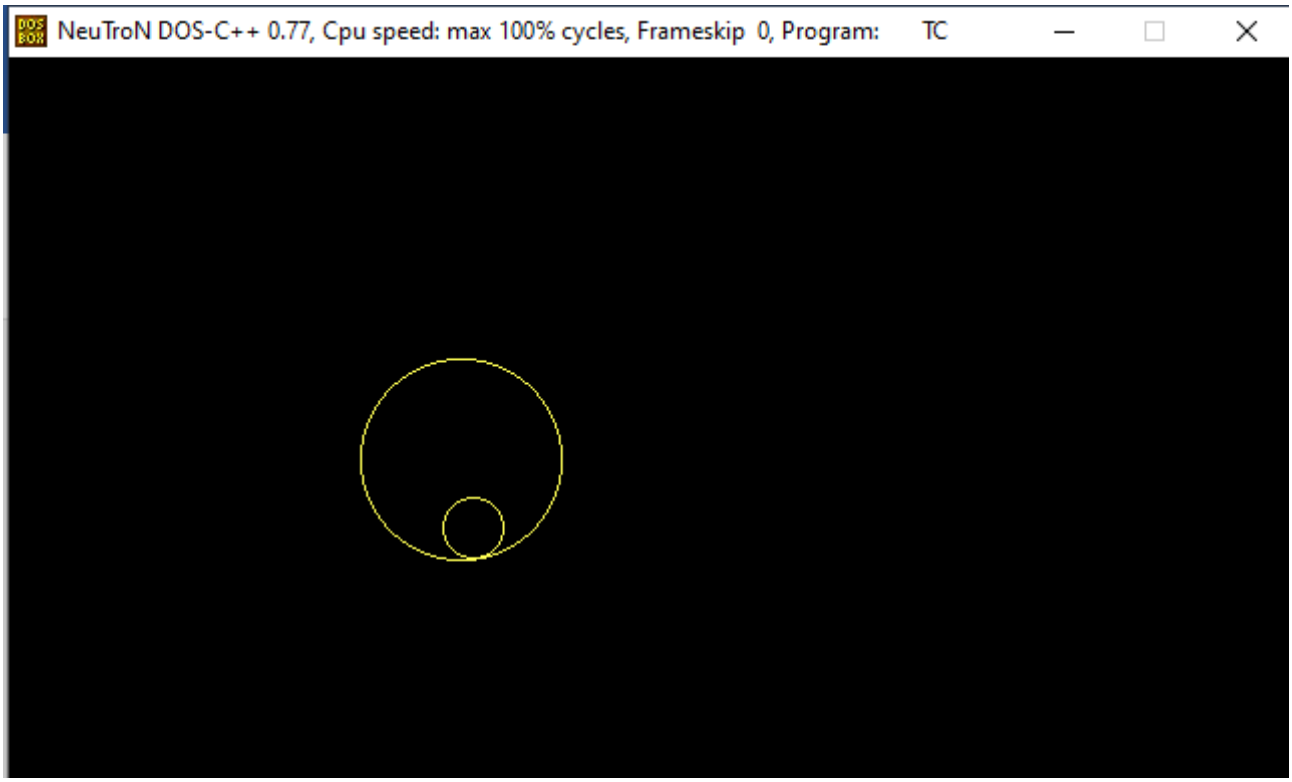
```
closegraph();
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

Q 22 - WAP To Rotate A Circle Inside Another Circle .

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
int xc=225,yc=200,r=50;
int x[3],y[3];
void drawcircles()
{
setcolor(YELLOW);
circle(xc,yc,r);
}
void main()
{
double angle=0,theta;
int i,a;
int gd=DETECT,gm;
initgraph(&gd,&gm,"..\\bgi");
a=0;
while(a<=100)
{
theta=M_PI*angle/180;
cleardevice();
drawcircles();
x[0]=xc+r*cos(theta); //x on circle
y[0]=yc+r*sin(theta); //y on circle
x[1]=xc+(r+15)*cos(theta);// x outside of circle
y[1]=yc+(r+15)*sin(theta);// y outside of circle
x[2]=xc+(r-15)*cos(theta);// x inside circle
y[2]=yc+(r-15)*sin(theta);// y inside circle
angle+=20;
circle(x[2],y[2],15); //for inner circle
a=a+1;
delay(50);
}
getch();
```

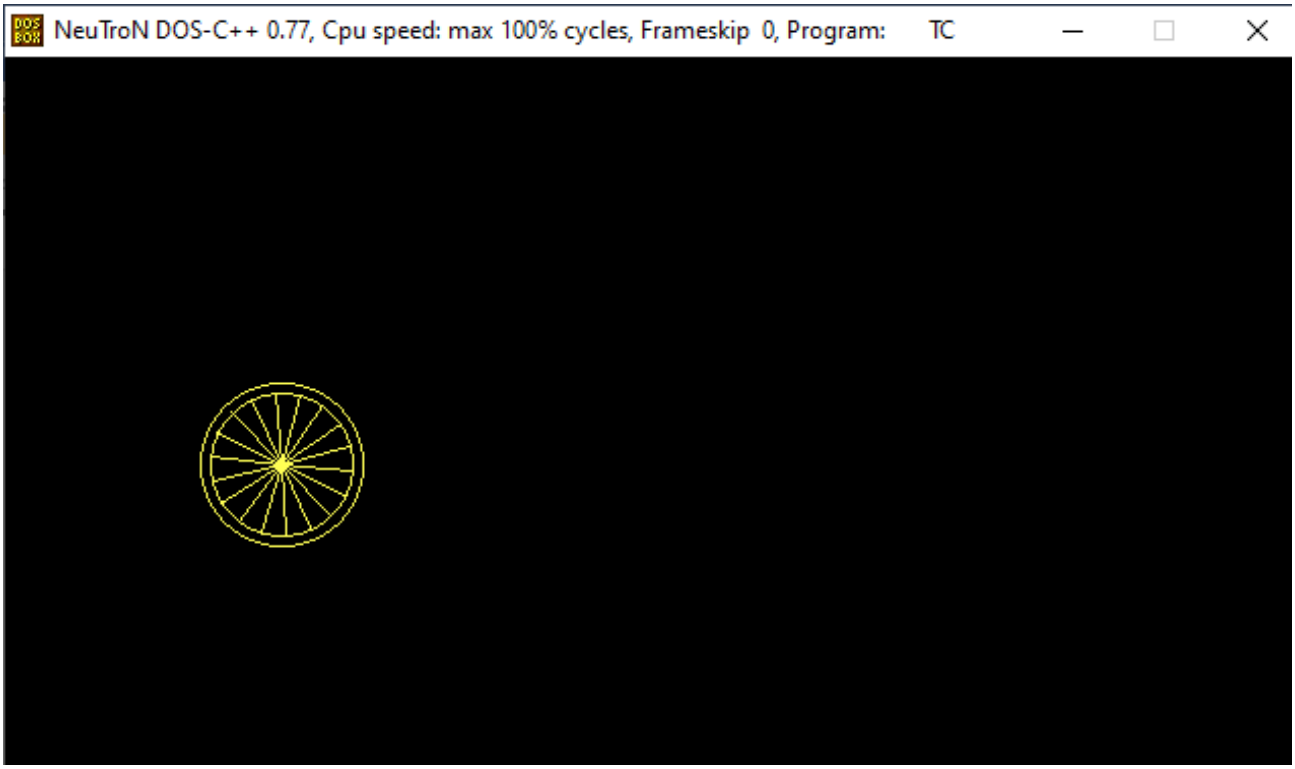
```
closegraph();
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

Q 23 - WAP To Make A Wheel.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
int xc=50,yc=200,r=35;
int x[15],y[15];
void drawcircles()
{
setcolor(YELLOW);
circle(xc,yc,r);
circle(xc,yc,r+5);
}
void main()
{ double angle=0,theta;
int i,a;
int gd=DETECT,gm;
initgraph(&gd,&gm,"..\\bgi");
a=xc+r;
while(!kbhit())
{
while(a<=630)
{
theta=M_PI*angle/180;
cleardevice();
drawcircles();
for(i=0;i<18;i++)
{
theta=M_PI*angle/180;
x[i]=xc+r*cos(theta);
y[i]=yc+r*sin(theta);
angle+=20;
line(xc,yc,x[i],y[i]);
}
angle+=2; xc+=2; a=xc+r;
delay(50);
```

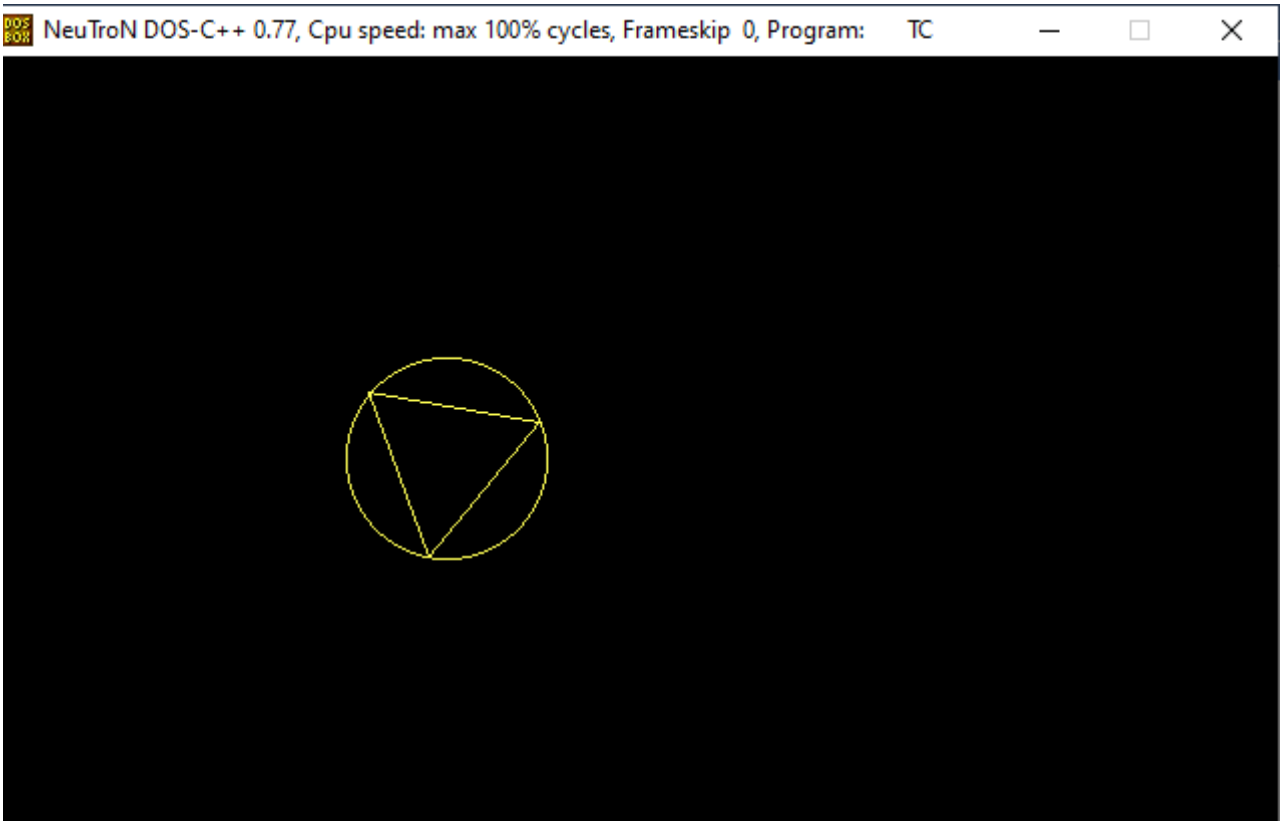
```
}
xc=50; r=35; a=xc+r;
}
getch();
closegraph();
}
```



Q 23 - WAP To Rotate The Triangle At Its Centre In Clockwise Direction.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
int xc=225,yc=200,r=50;
int x[3],y[3];
void drawcircles()
{
setcolor(YELLOW);
circle(xc,yc,r);
line(x[0],y[0],x[1],y[1]);
line(x[1],y[1],x[2],y[2]);
line(x[2],y[2],x[0],y[0]);
}
void main()
{
double angle=0,theta,ang;
ang = M_PI*120/180;
int i,a;
int gd=DETECT,gm;
initgraph(&gd,&gm,"..\\bgi");
a=0;
while(a<=100)
{
theta=M_PI*angle/180;
cleardevice();
drawcircles();
x[0]=xc+r*cos(theta);
y[0]=yc+r*sin(theta);
x[1]=xc+r*cos(theta+ang);
y[1]=yc+r*sin(theta+ang);
x[2]=xc+r*cos(theta+2*ang);
y[2]=yc+r*sin(theta+2*ang);
angle+=20;
a=a+1;
```

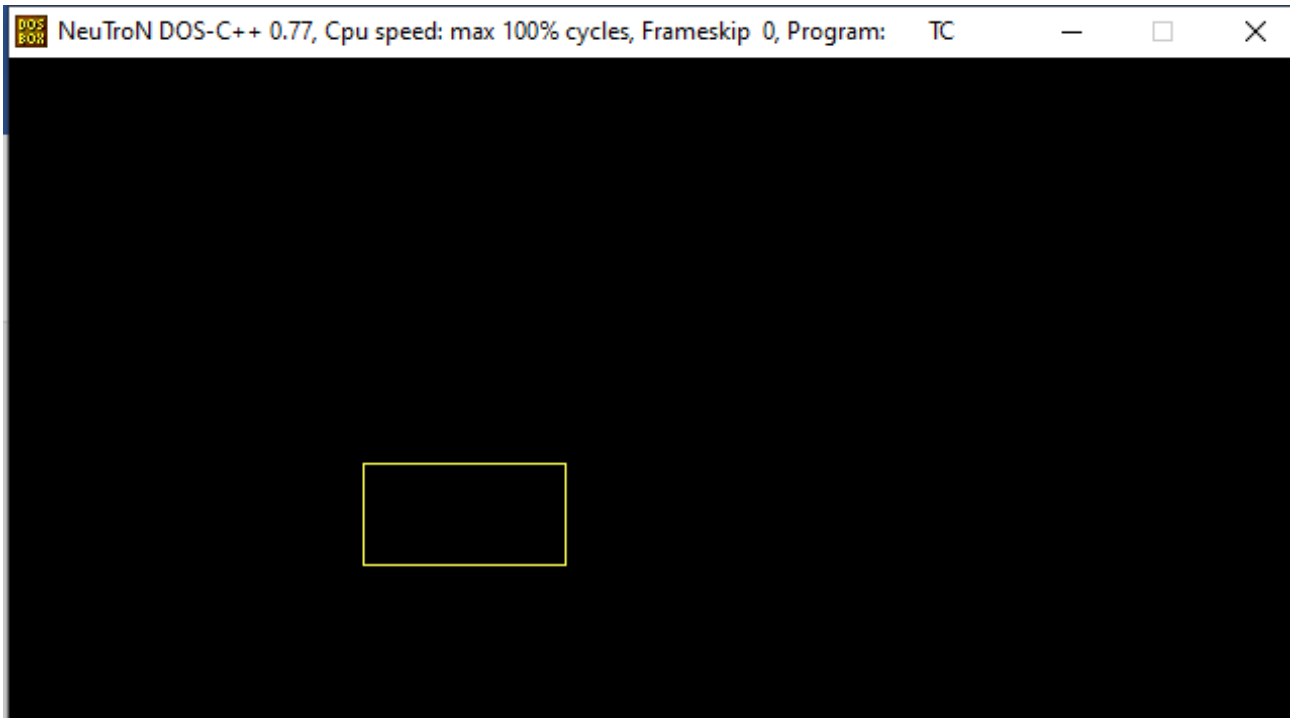
```
delay(50);
}
getch();
closegraph();
}
```

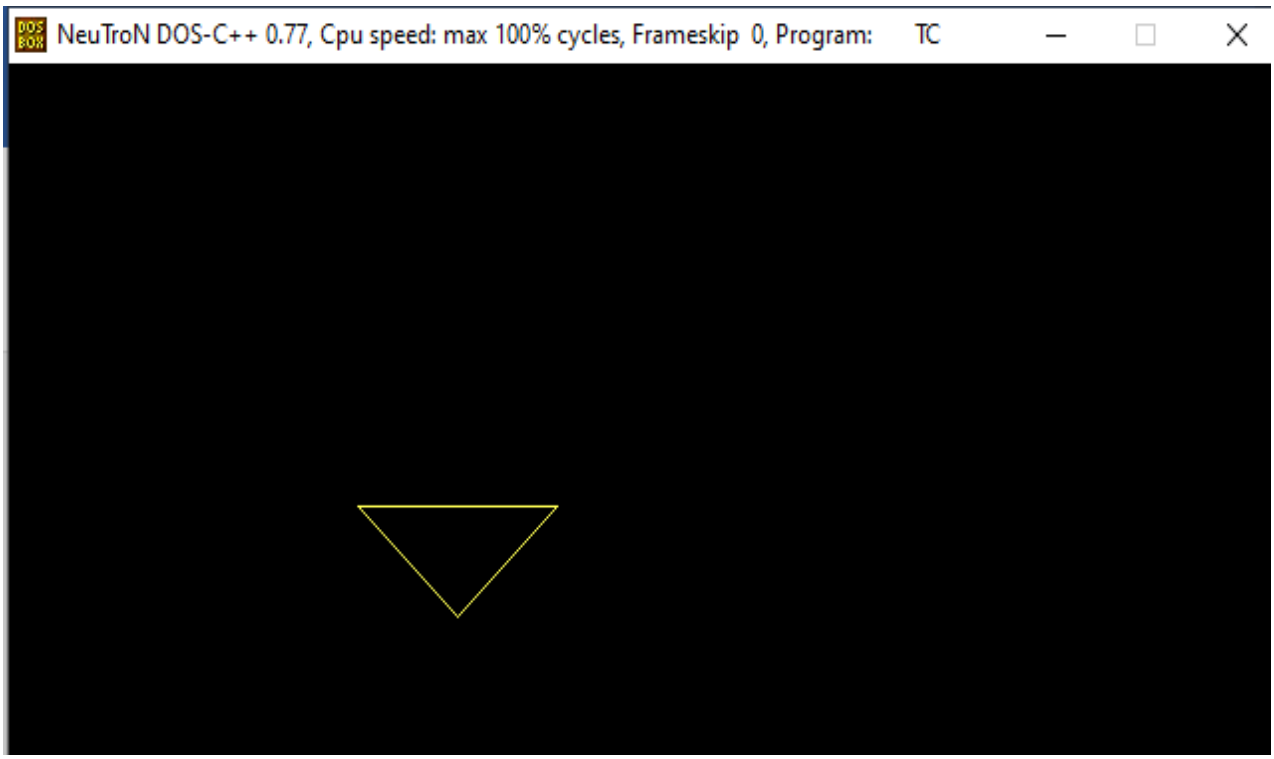


Q 24 - WAP To Change The Triangle In To A Rectangle.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
int xc=225,yc=200,r=50;
int x[3],y[3];
void draw()
{
setcolor(YELLOW);
line(x[0],y[0],x[1],y[1]);
line(x[1],y[1],x[2],y[2]);
line(x[2],y[2],x[0],y[0]);
delay(200);
cleardevice();
line(x[0],y[0],x[1],y[1]);
line(x[0],y[0],x[0],y[0]+r);
line(x[1],y[1],x[1],y[1]+r);
line(x[0],y[0]+r,x[1],y[1]+r);
}
void main()
{
double angle=0,theta,ang;
ang = M_PI/180;
int i,a;
int gd=DETECT,gm;
initgraph(&gd,&gm,"..\\bgi");
a=0;
while(a<=10)
{
theta=M_PI*angle/180;
cleardevice();
x[0]=xc+r*cos(theta);
y[0]=yc+r*sin(theta);
x[1]=xc+r*cos(theta+ang*180);
y[1]=yc+r*sin(theta+ang*180);
x[2]=xc+r*cos(theta+ang*90);
y[2]=yc+r*sin(theta+ang*90);
draw();
a = a+1;
delay(200);
}
getch();
closegraph();
```

```
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

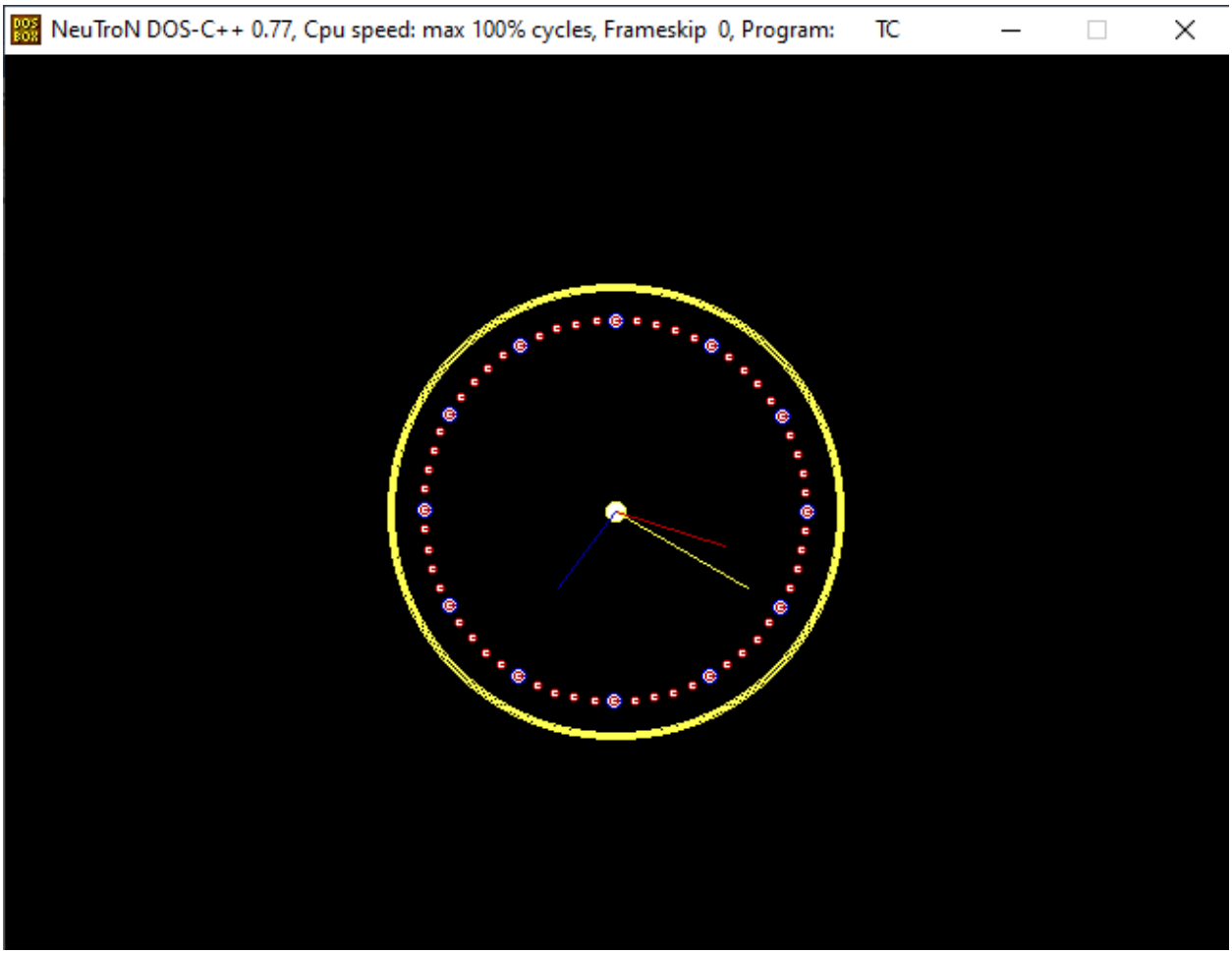

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

Q26- WAP TO MAKE A ANALOG CLOCK

```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
#include<math.h>
#include<dos.h>
#include<time.h>
 #define PI 3.147
void clockLayout();
void secHand();
void hrHand();
void minHand();
int maxx,maxy;
void main()
{int gdriver=DETECT,gmode,error;
 initgraph(&gdriver,&gmode,"c:\\turboc3\\bgi");
 error=graphresult();
 if(error!=grOk)
 {printf("Error in graphics, code=
%d",grapherrormsg(error));
  exit(0);
 }
  while(1)
 { clockLayout();
  secHand();
  minHand();
  hrHand();
  sleep(1);
  cleardevice();
 }
}void clockLayout()
{
 int i,x,y,r;
 float j;
 maxx=getmaxx();
 maxy=getmaxy();
 for(i=1;i<5;i++)
 {
  setcolor(YELLOW);
  circle(maxx/2,maxy/2,120-i);
```

```
 }

 pieslice(maxx/2,maxy/2,0,360,5);
 x=maxx/2+100;y=maxy/2;
 r=100;
 setcolor(BLUE);

 for(j=PI/6;j<=(2*PI);j+=(PI/6))
 {
  pieslice(x,y,0,360,4);
  x=(maxx/2)+r*cos(j);
  y=(maxy/2)+r*sin(j);
 }

 x=maxx/2+100;y=maxy/2;
 r=100;
 setcolor(RED);

 for(j=PI/30;j<=(2*PI);j+=(PI/30))
 {
  pieslice(x,y,0,360,2);
  x=(maxx/2)+r*cos(j);
  y=(maxy/2)+r*sin(j);
 }
}

void secHand()
{
 struct time t;
 int r=80,x=maxx/2,y=maxy/2,sec;
 float O;

 maxx=getmaxx();maxy=getmaxy();
 gettime(&t);
 sec=t.ti_sec;
 O=sec*(PI/30)-(PI/2);
 setcolor(YELLOW);
 line(maxx/2,maxy/2,x+r*cos(O),y+r*sin(O));
}
```

```
void hrHand()
{
 int r=50,hr,min;
 int x,y;
 struct time t;
 float O;
 maxx=getmaxx();
 maxy=getmaxy();
 x=maxx/2,y=maxy/2;
 gettime(&t);
 hr=t.ti_hour;
 min=t.ti_min;
 if(hr<=12)O=(hr*(PI/6)-(PI/2))+((min/12)*(PI/30));
 if(hr>12) O=((hr-12)*(PI/6)-
(PI/2))+((min/12)*(PI/30));
 setcolor(BLUE);
 line(maxx/2,maxy/2,x+r*cos(O),y+r*sin(O));
}
void minHand()
{ int r=60,min;
 int x,y;
 float O;
 struct time t;
 maxx=getmaxx();
 maxy=getmaxy();
 x=maxx/2;
 y=maxy/2;
 gettime(&t);
 min=t.ti_min;
 O=(min*(PI/30)-(PI/2));
 setcolor(RED);
 line(maxx/2,maxy/2,x+r*cos(O),y+r*sin(O));
}
```


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC