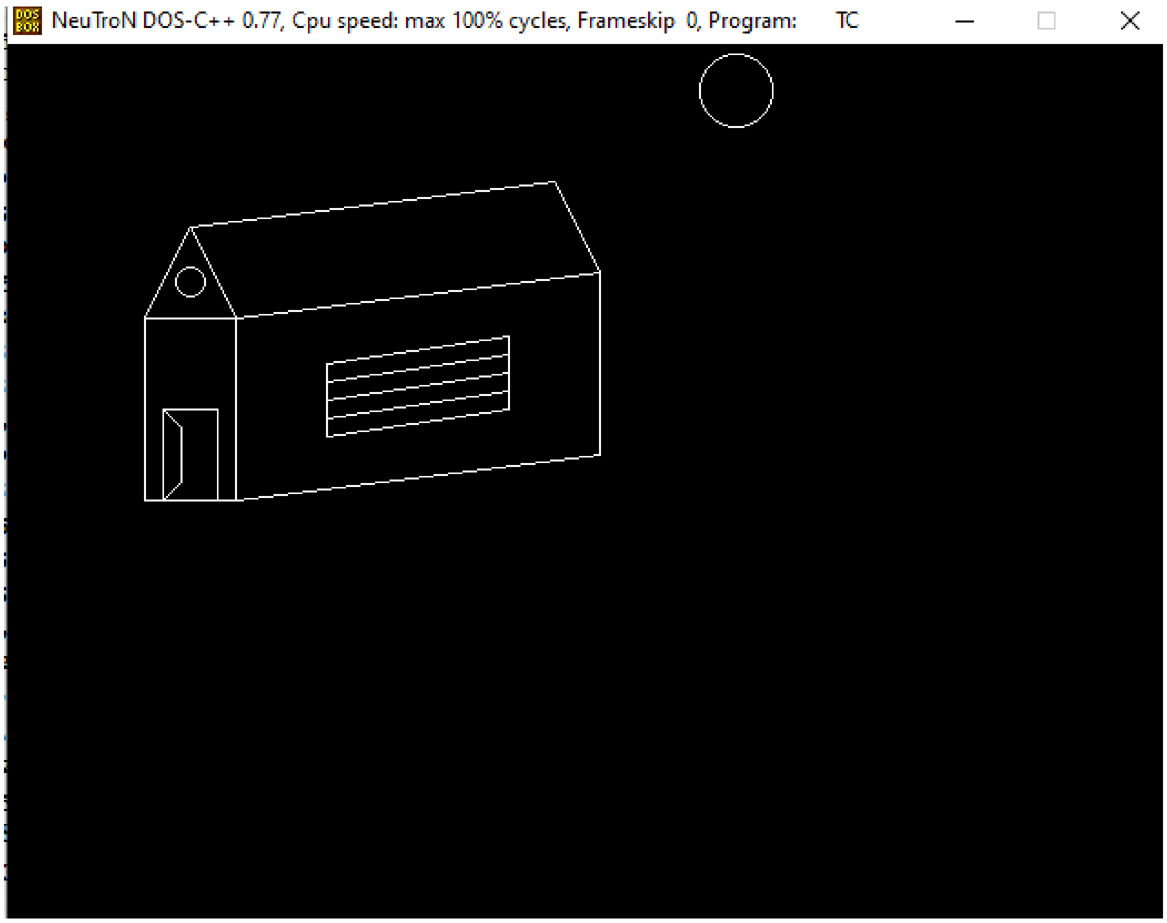
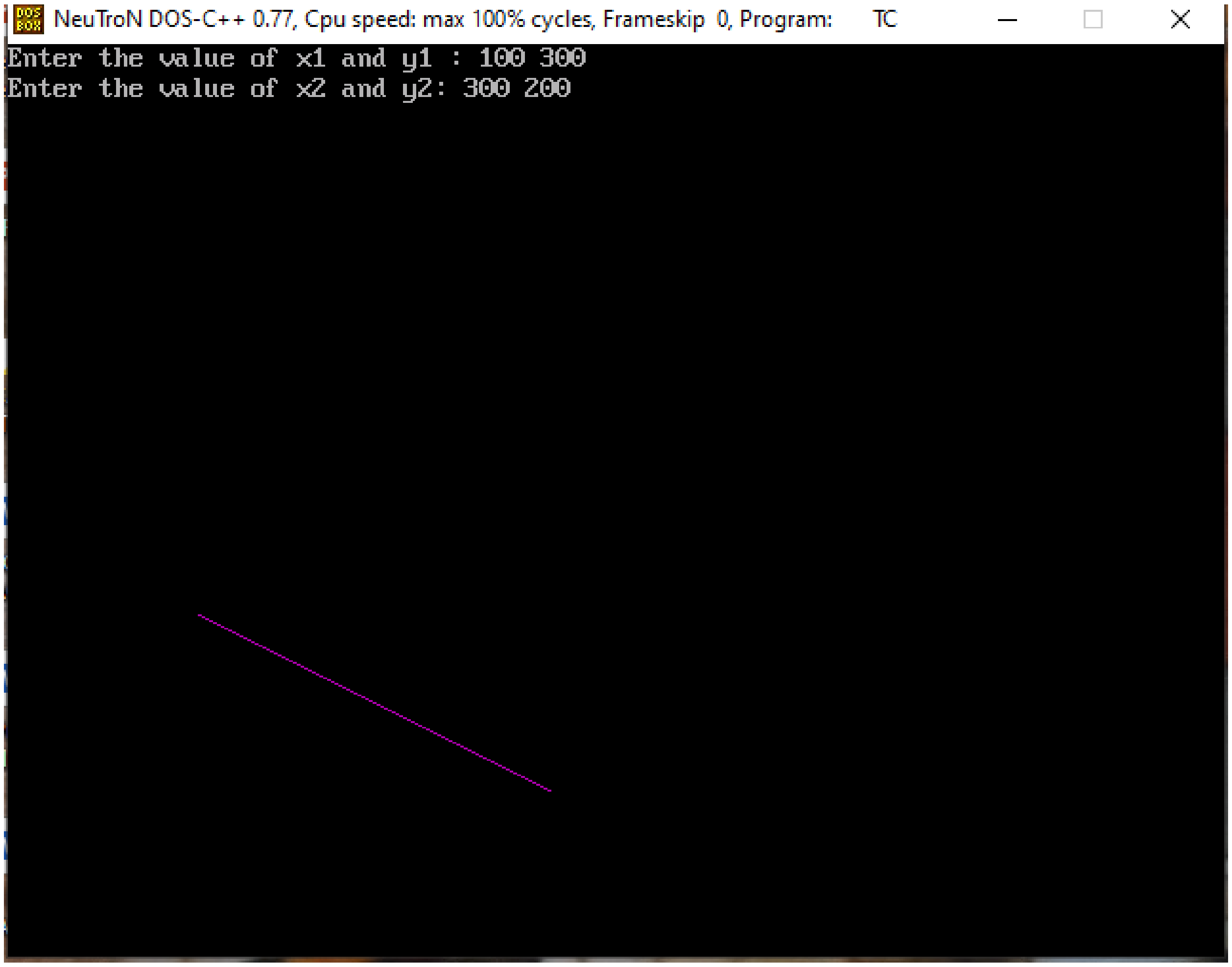
1. WAP to Draw Objects (HUT/SMILY Face)

```
#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);

  CIRCLE(100,130,8);
  GETCH(); }
```


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

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

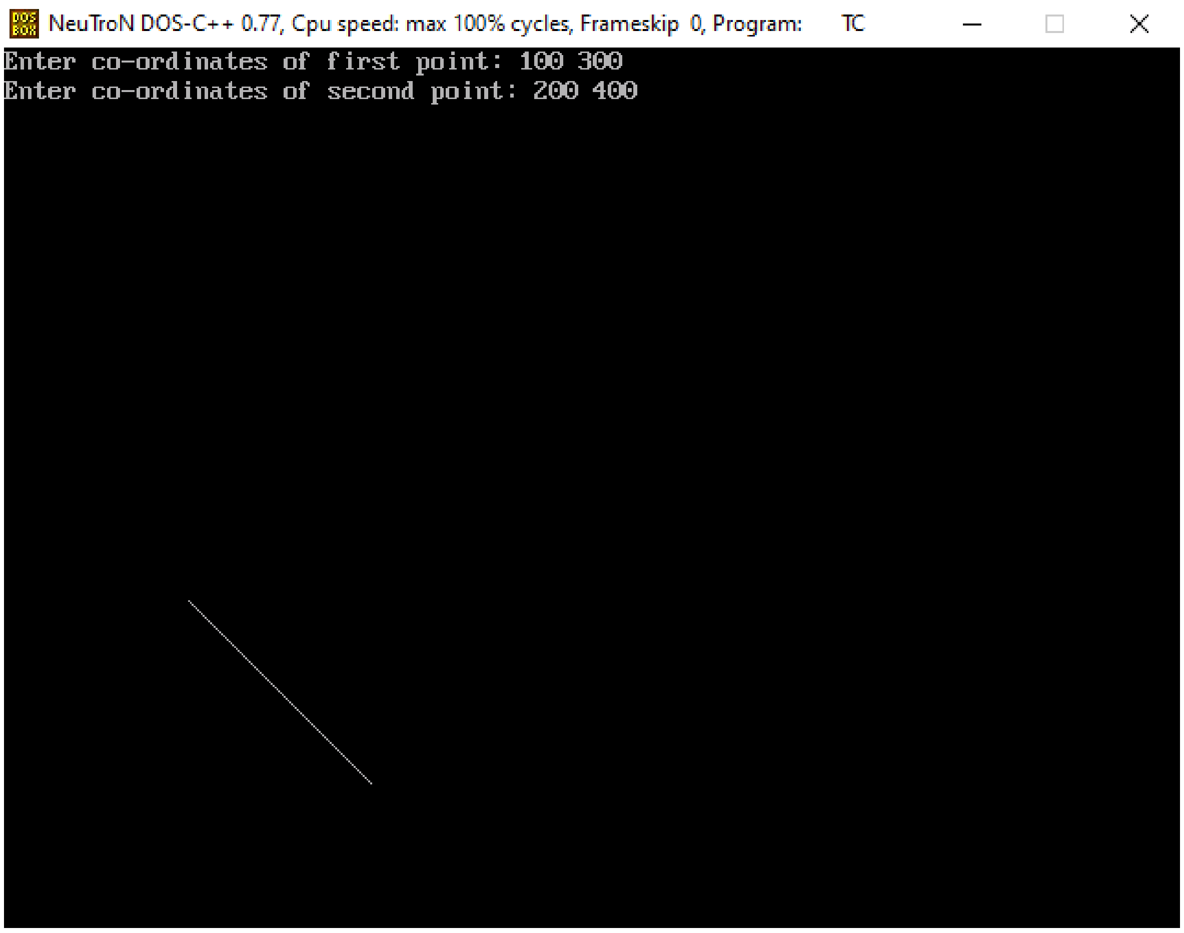
```
#INCLUDE<GRAPHICS.H>
#INCLUDE<IOSTREAM.H>
#INCLUDE<MATH.H>
#INCLUDE<DOS.H>
INT MAIN( ){
FLOAT X,Y,X1,Y1,X2,Y2,DX,DY,STEP;
INT  I,GD=DETECT,GM;
INITGRAPH(&GD,&GM,"C:\\TURBOC3\\BGI");
COUT<<"ENTER THE   VALUE  OF  X1  AND  Y1 : ";
CIN>>X1>>Y1;
COUT<<"ENTER THE   VALUE  OF  X2  AND  Y2: ";
CIN>>X2>>Y2;
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
```

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

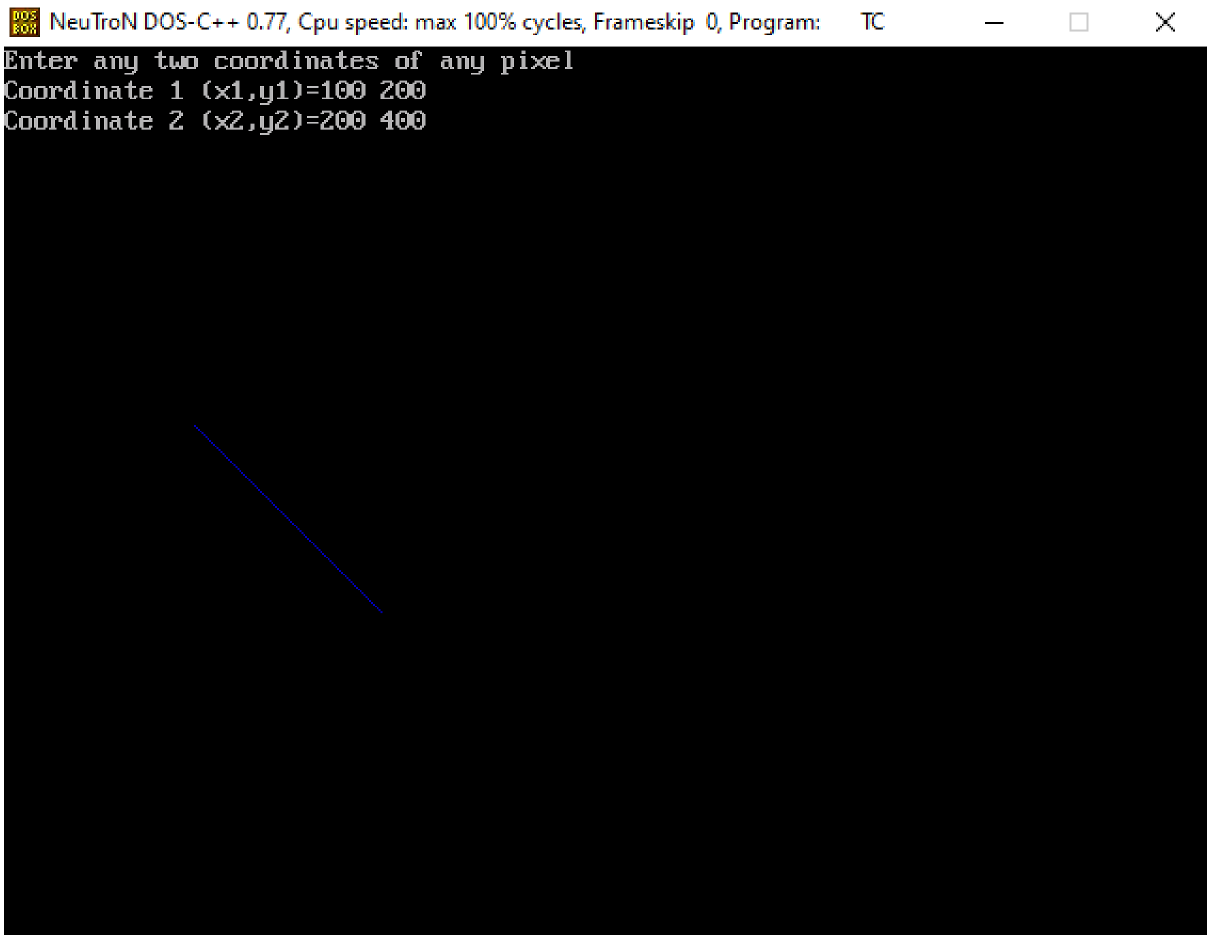
```
#INCLUDE<IOSTREAM.H>
#INCLUDE<GRAPHICS.H>
VOID DRAWLINE(INT X0, INT Y0, INT X1, INT Y1){INT DX, DY,
P, X, Y;
DX=X1-X0;
DY=Y1-Y0;
X=X0; Y=Y0;
P=2*DY-DX;
WHILE(X<X1){
IF(P>=0){

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
```

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;}
```

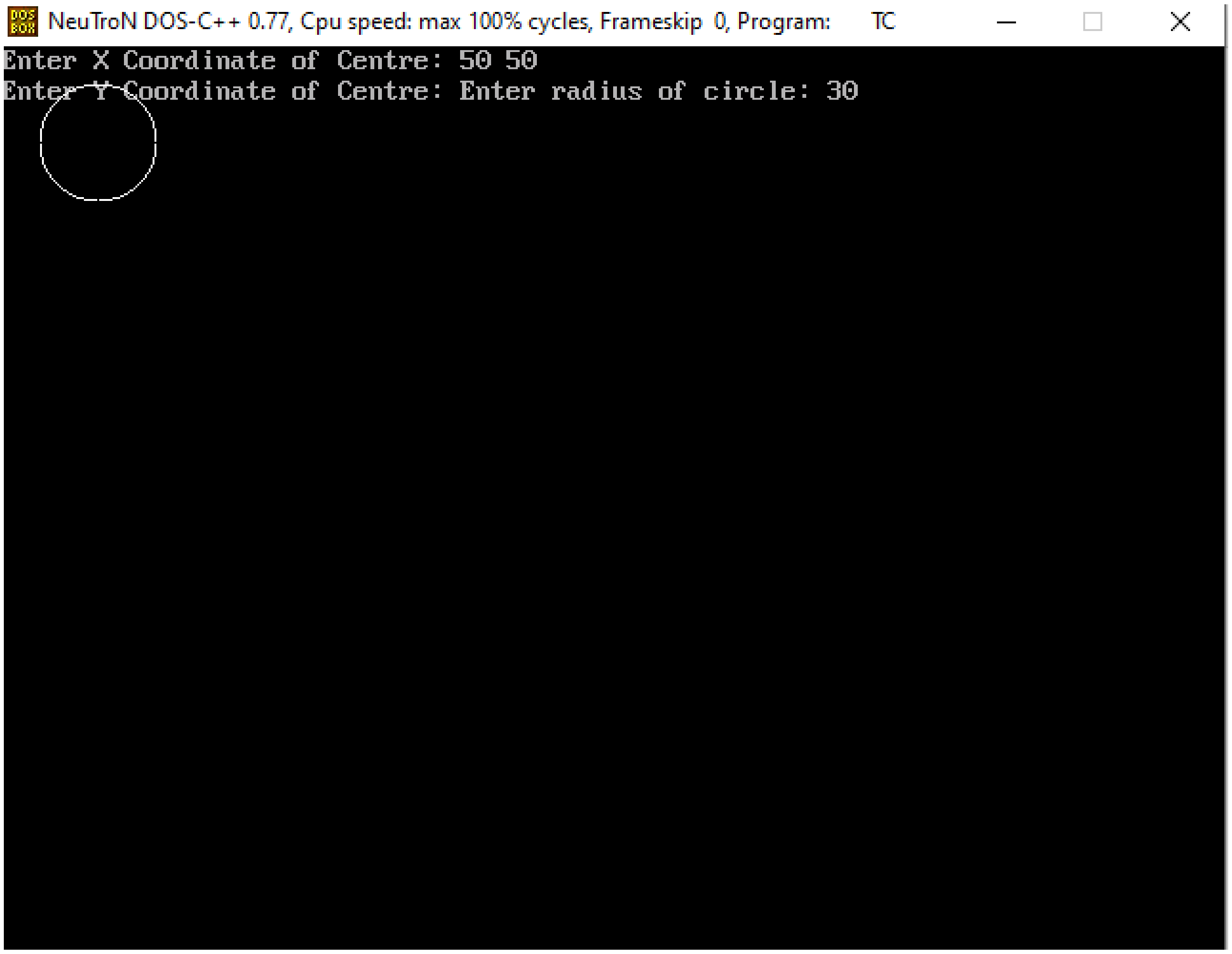


NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter any two coordinates of any pixel
Coordinate 1 (x1,y1)=100 200
Coordinate 2 (x2,y2)=200 400
```

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

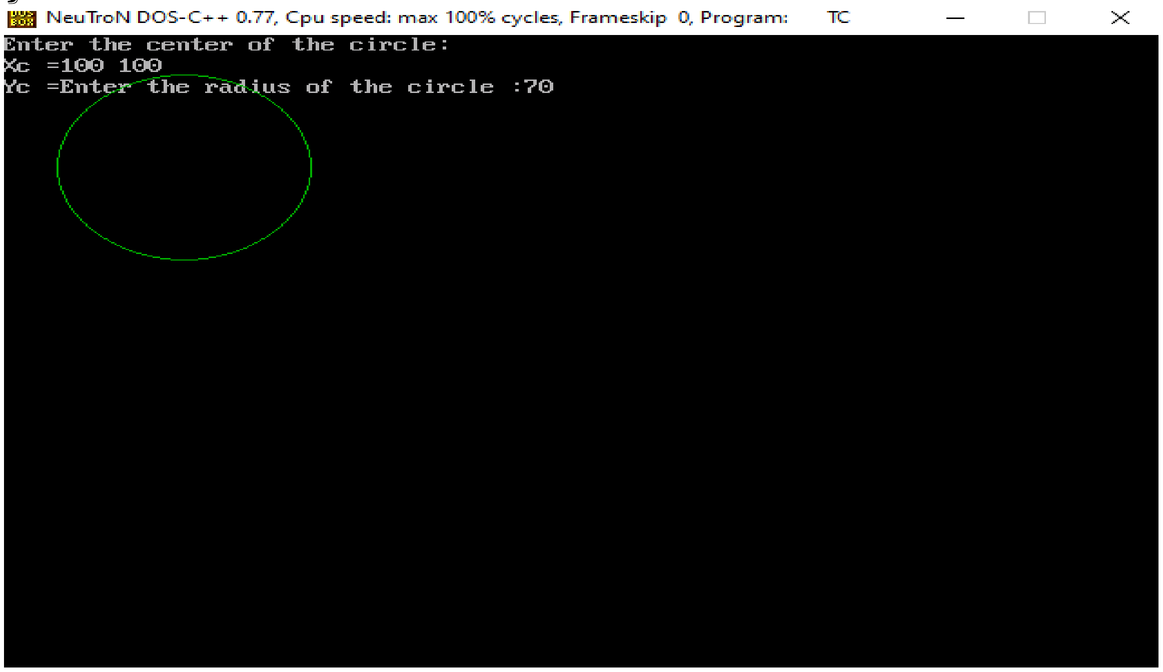
```
#INCLUDE<GRAPHICS.H>
#INCLUDE<CONIO.H>
#INCLUDE<STDIO.H>
#INCLUDE<MATH.H>
VOID MAIN(){
INT GD=DETECT,GM;
INT X,Y,A,B,RADIUS,D,FLAG;
INITGRAPH(&GD,&GM,"C:\\TURBOC3\\BGI\\");
PRINTF("ENTER X COORDINATE OF CENTRE: ");
SCANF("%D",&A);
PRINTF("ENTER Y COORDINATE OF CENTRE: ");
SCANF("%D",&B);
PRINTF("ENTER 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
```

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

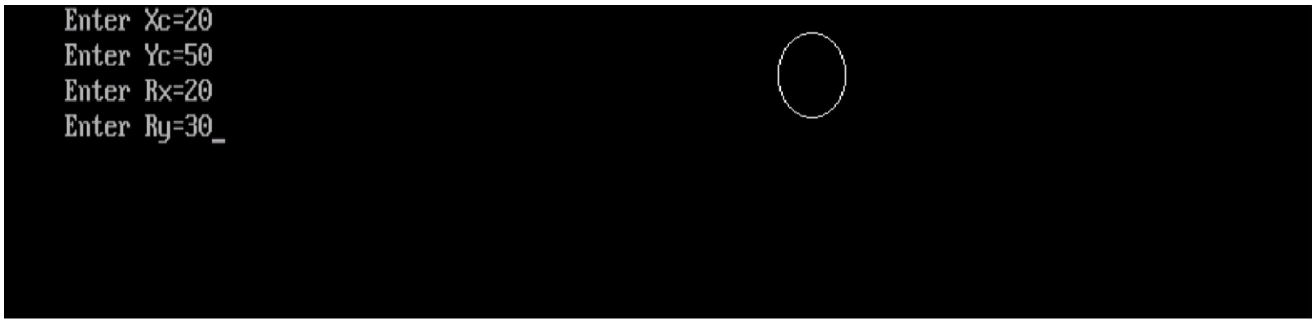
```
#INCLUDE<DOS.H>
#INCLUDE<STDIO.H>
#INCLUDE<CONIO.H>
#INCLUDE<GRAPHICS.H>
VOID DRAW_CIRCLE(INT,INT,INT); VOID
SYMMETRY(INT,INT,INT,INT);VOID
MAIN()
{ INT XC,YC,R; INT
GD=DETECT,GM;
INITGRAPH(&GD,&GM,"C:\\TURBOC3\\BGI");
PRINTF("ENTER THE CENTER OF THE CIRCLE:\N");
PRINTF("XC =");
SCANF("%D",&XC);
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);
}
```


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

```
Enter the center of the circle:
Xc =100 100
Yc =Enter the radius of the circle :70
```

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

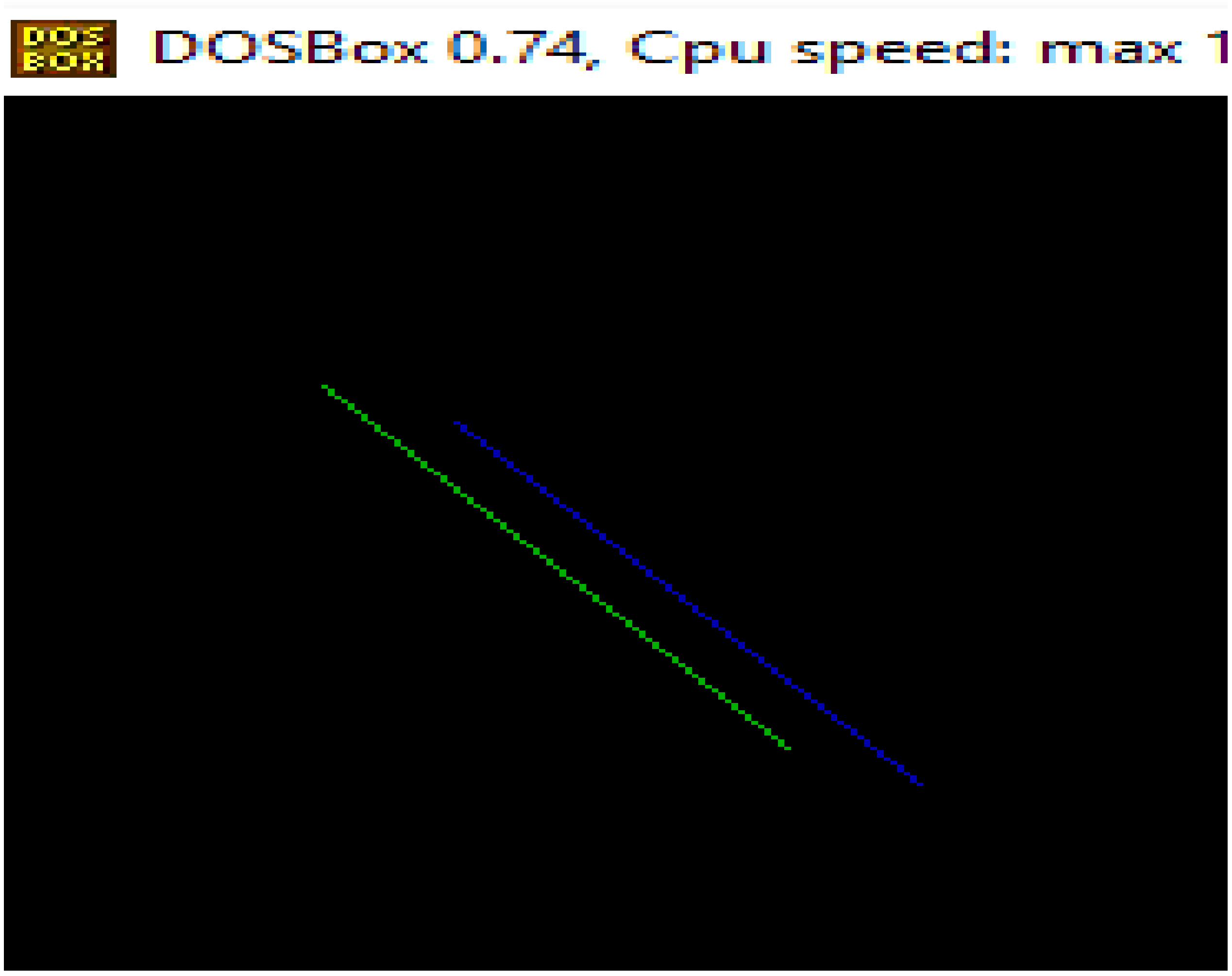
```
#INCLUDE<STDIO.H>
#INCLUDE<CONIO.H>
#INCLUDE<GRAPHICS.H>
VOID ELLIPSE(INT XC,INT YC,INT RX,INT RY){INT GM=DETECT,GD;
INT X, Y, P;
CLRSCR(); INITGRAPH(&GM,&GD,"C:\\TC\\BGI");
X=0;
Y=RY;
P=(RY*RY)-(RX*RX*RY)+((RX*RX)/4);WHILE((2*X*RY*RY)<(2*Y*RX*RX))
{
PUTPIXEL(XC+X,YC-Y,WHITE);
PUTPIXEL(XC-X,YC+Y,WHITE);
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();
}
```



```
Enter Xc=20
Enter Yc=50
Enter Rx=20
Enter Ry=30_
```

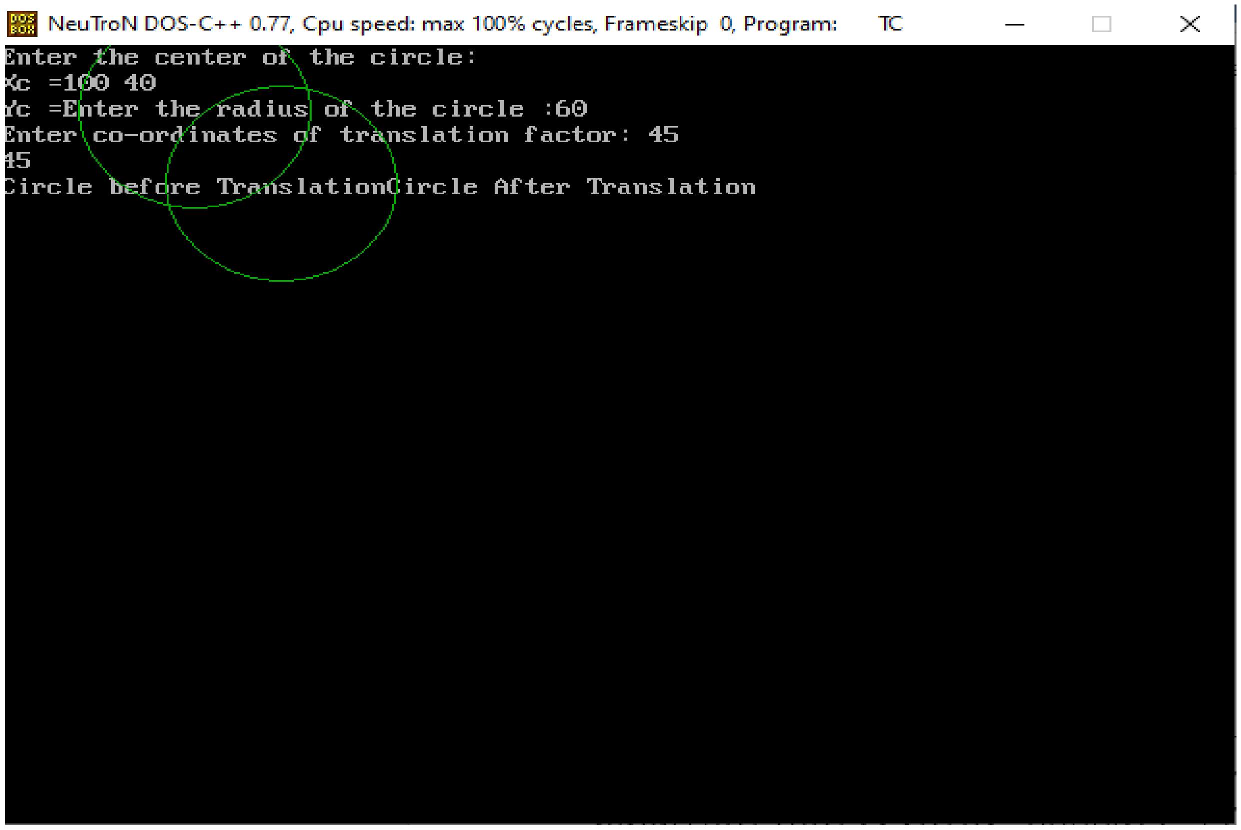
8 WAP for Translation in 2 – Dimension.

```
#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

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)
}
```


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

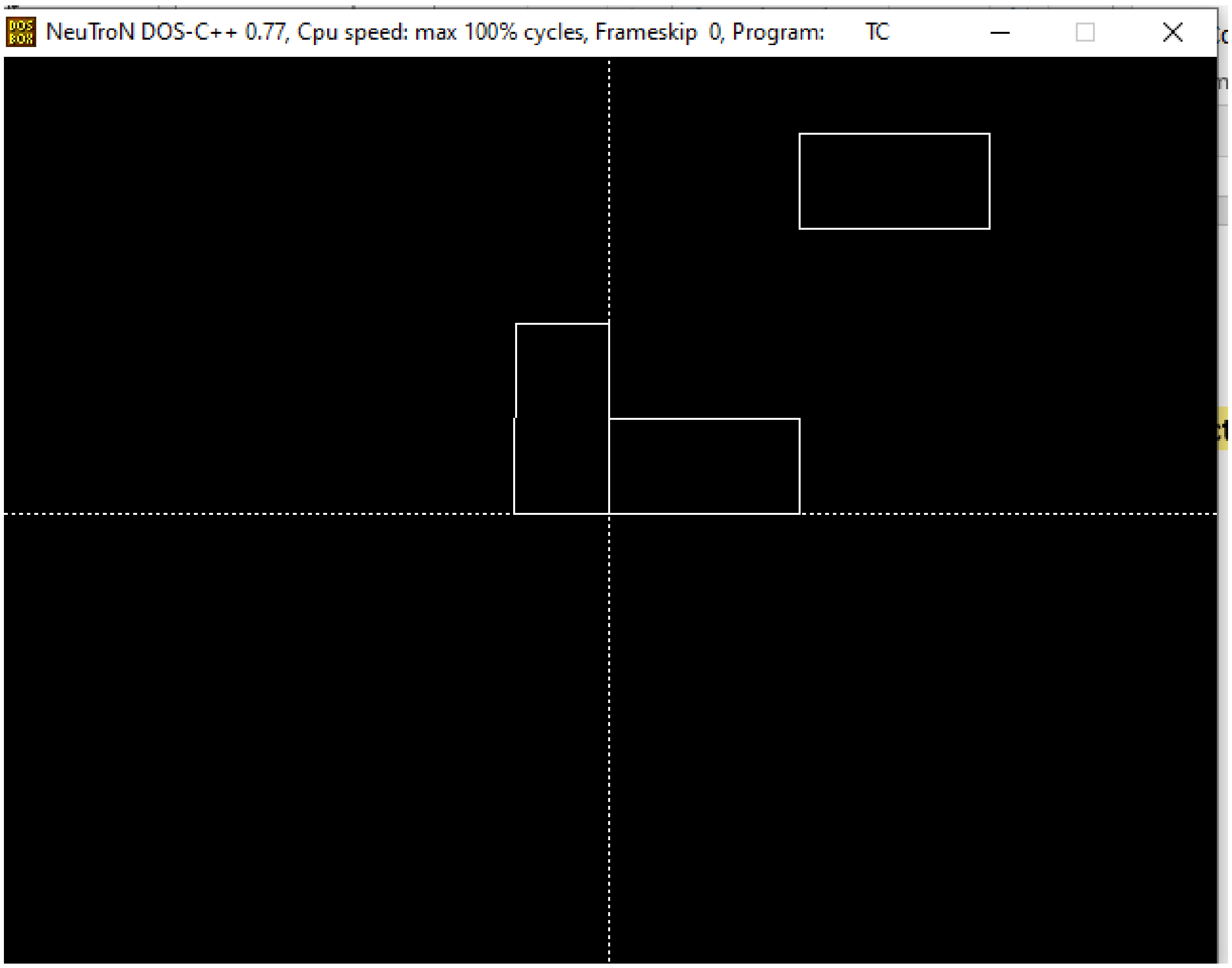
```
Enter the center of the circle:
Xc =100 40
Yc =Enter the radius of the circle :60
Enter co-ordinates of translation factor: 45
45
Circle before TranslationCircle After Translation
```

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){
```

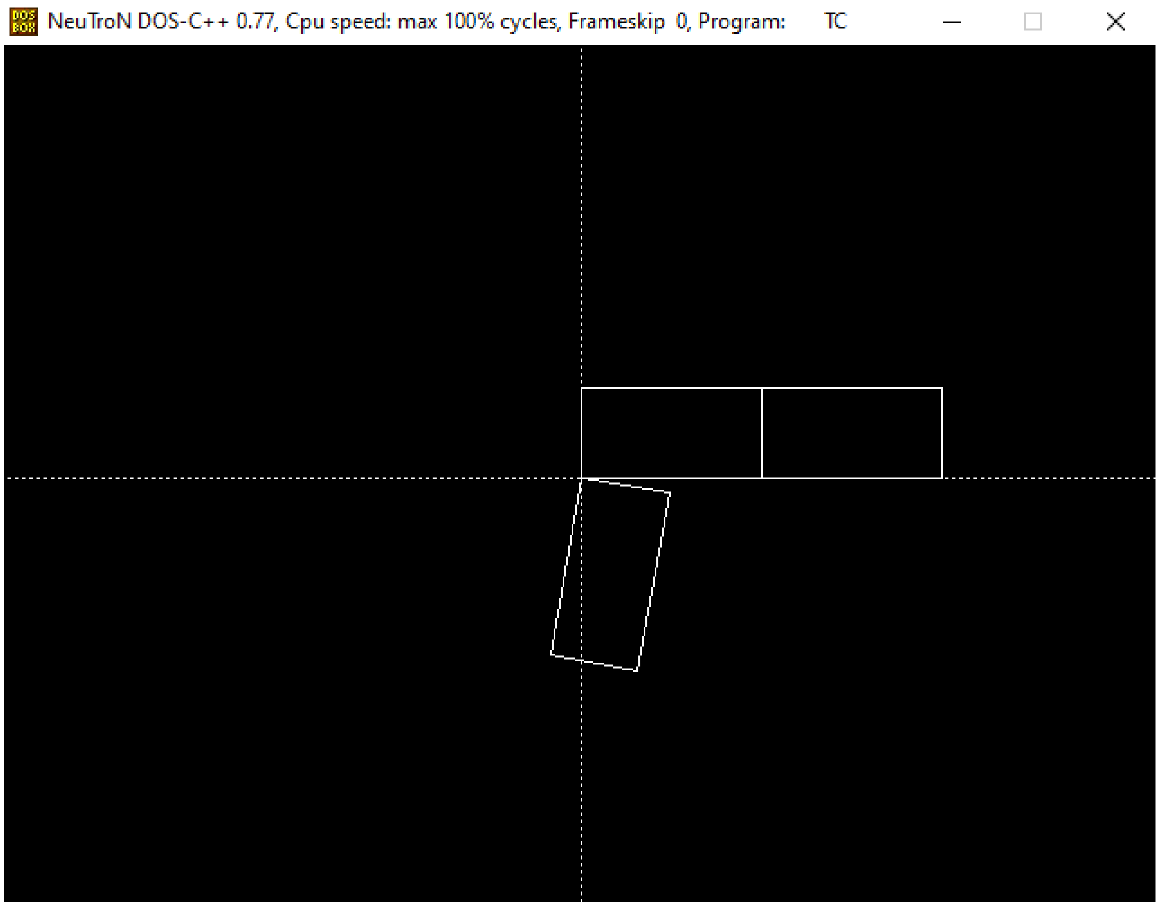
```
CASE 1: PRINTF("ENTER NEGATIVE ANGLE FOR CLOCKWISEROTATION
");
SCANF("%D",&ANGLE);
PRINTF("ENTER THE X- AND Y-COORDINATES OF THE POINT:");
SCANF("%D%D",&X,&Y);
INI(); CLEARDEVICE();
RESET(R);
TRANSLATE(R,X,Y);
DRAW(R);
PUTPIXEL(320+X,240-Y,WHITE);
GETCH();
TRANSLATE(R,-X,-Y);
DRAW(R);GETCH();
ROTATE(R,ANGLE);
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

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

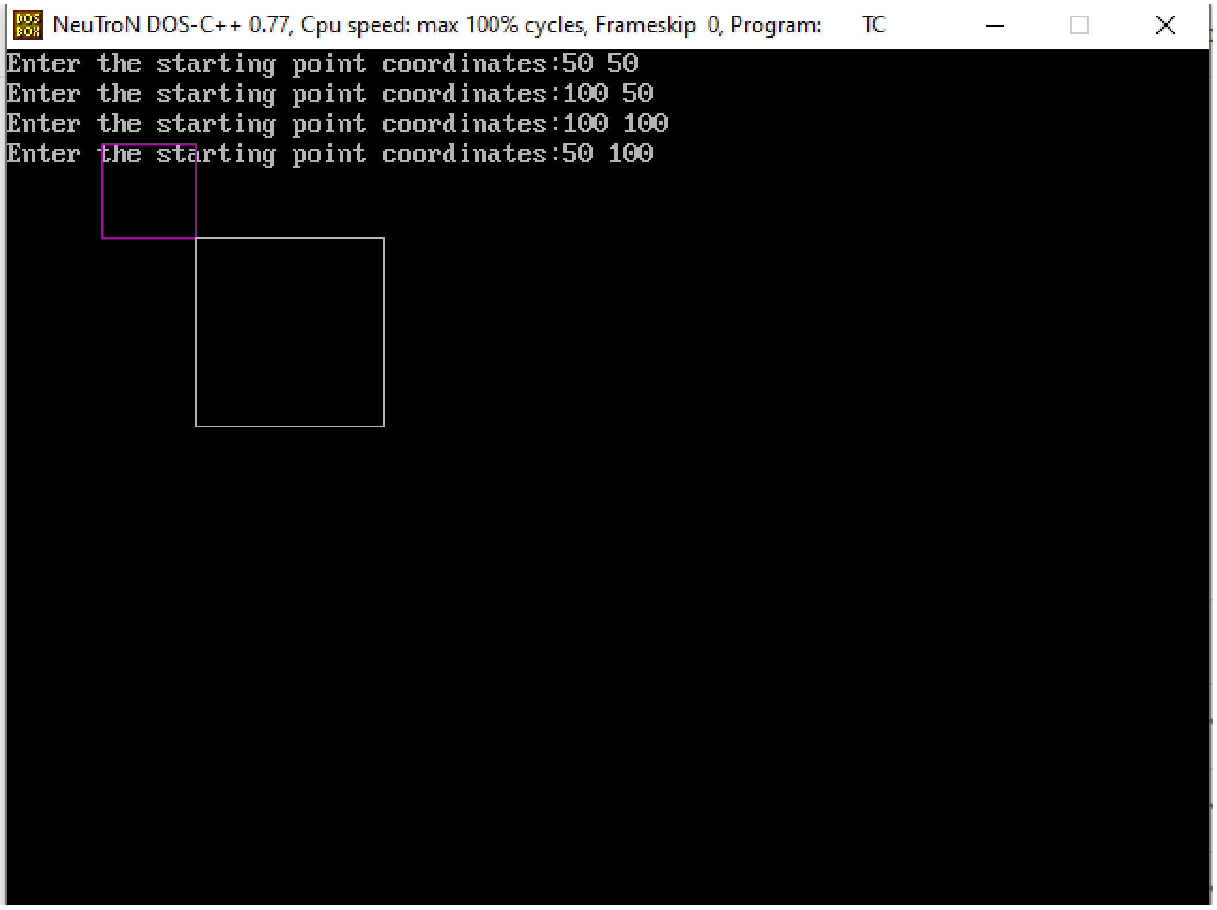
```
#INCLUDE<STDIO.H>
#INCLUDE<GRAPHICS.H>
#INCLUDE<STDLIB.H>
#INCLUDE<MATH.H>
#INCLUDE<IOSTREAM.H>
#INCLUDE<CONIO.H> VOID
DRAW (INT R[][2])
{ INT I;
SETLINESTYLE (DOTTED_LINE, 0, 1);
LINE (320, 0, 320, 480);
LINE (0, 240, 640, 240);
SETLINESTYLE (SOLID_LINE, 0, 1);
LINE (320+R[0][0],          240-R[0][1],      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

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

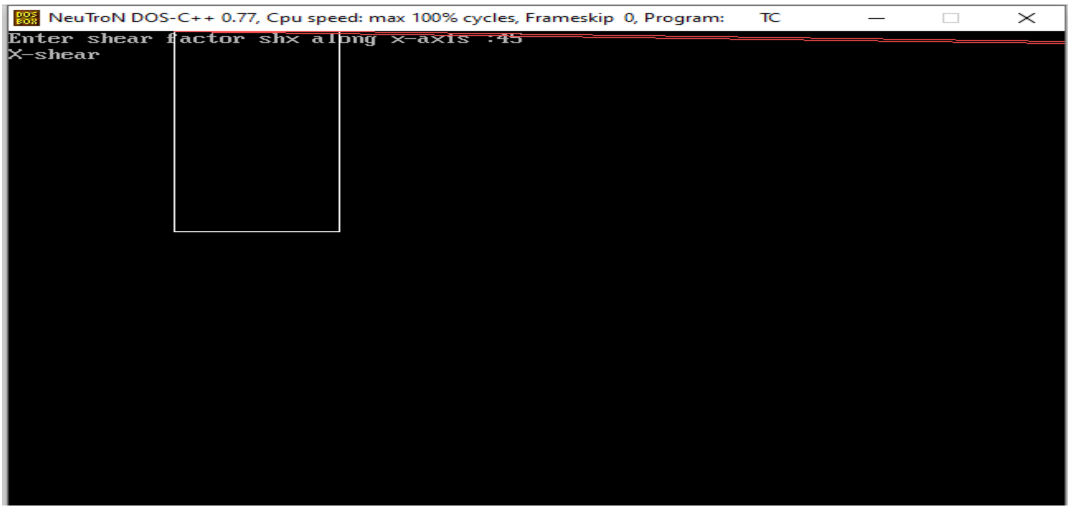
```
#INCLUDE<STDIO.H>
#INCLUDE<CONIO.H>
#INCLUDE<GRAPHICS.H>
#INCLUDE<MATH.H> VOID
MAIN()
{
INT GD=DETECT,GM;
FLOAT X[4],Y[4],SX=2,SY=2;INT
I;
INITGRAPH(&GD,&GM,"C:\\TURBOC3\\BGI");
FOR(I=0;I<4;I++)
{
PRINTF("ENTER THE STARTING POINT COORDINATES:");SCANF("%F
%F",&X[I],&Y[I]);
}
SETCOLOR(5);
LINE(X[0],Y[0],X[1],Y[1]);
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
```

13. WAP To Shear A Square In X-Direction.

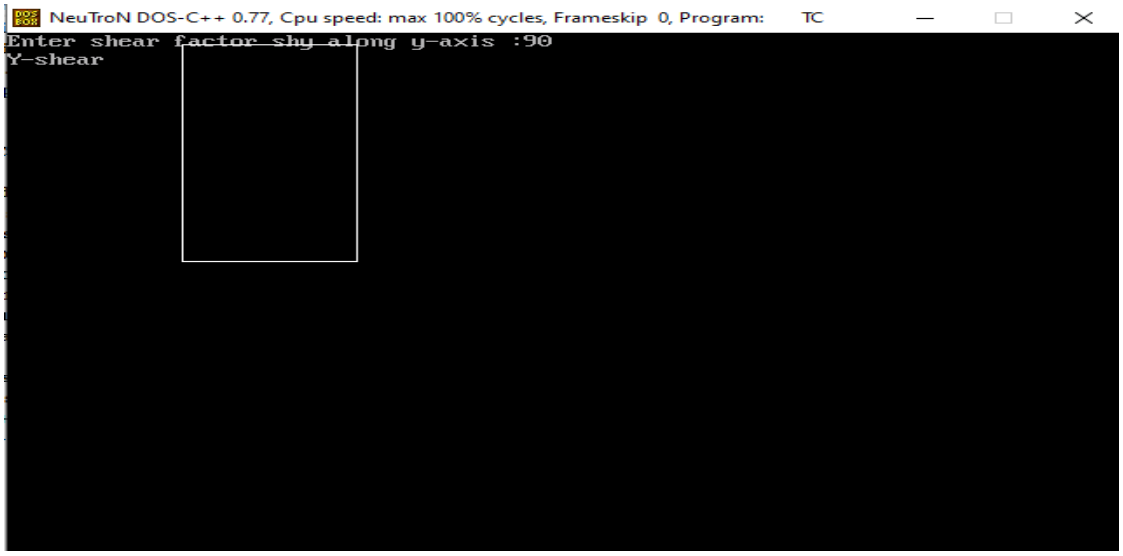
```
#INCLUDE<STDIO.H>
#INCLUDE<CONIO.H>
#INCLUDE<DOS.H>
#INCLUDE<GRAPHICS.H>
VOID MAIN()
{
INT GD=DETECT,GM;
FLOAT SHX,SHY;
INITGRAPH(&GD,&GM,"C:\\TURBOC3\\BGI"); PRINTF("ENTER
SHEAR FACTOR SHX ALONG X-AXIS :");SCANF("%F",&SHX);
LINE(100,0,200,0);
LINE(200,0,200,200);
LINE(200,200,100,200);

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();
}
```


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

```
Enter shear factor shx along x-axis :45
X-shear
```

14. 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();
}
```


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

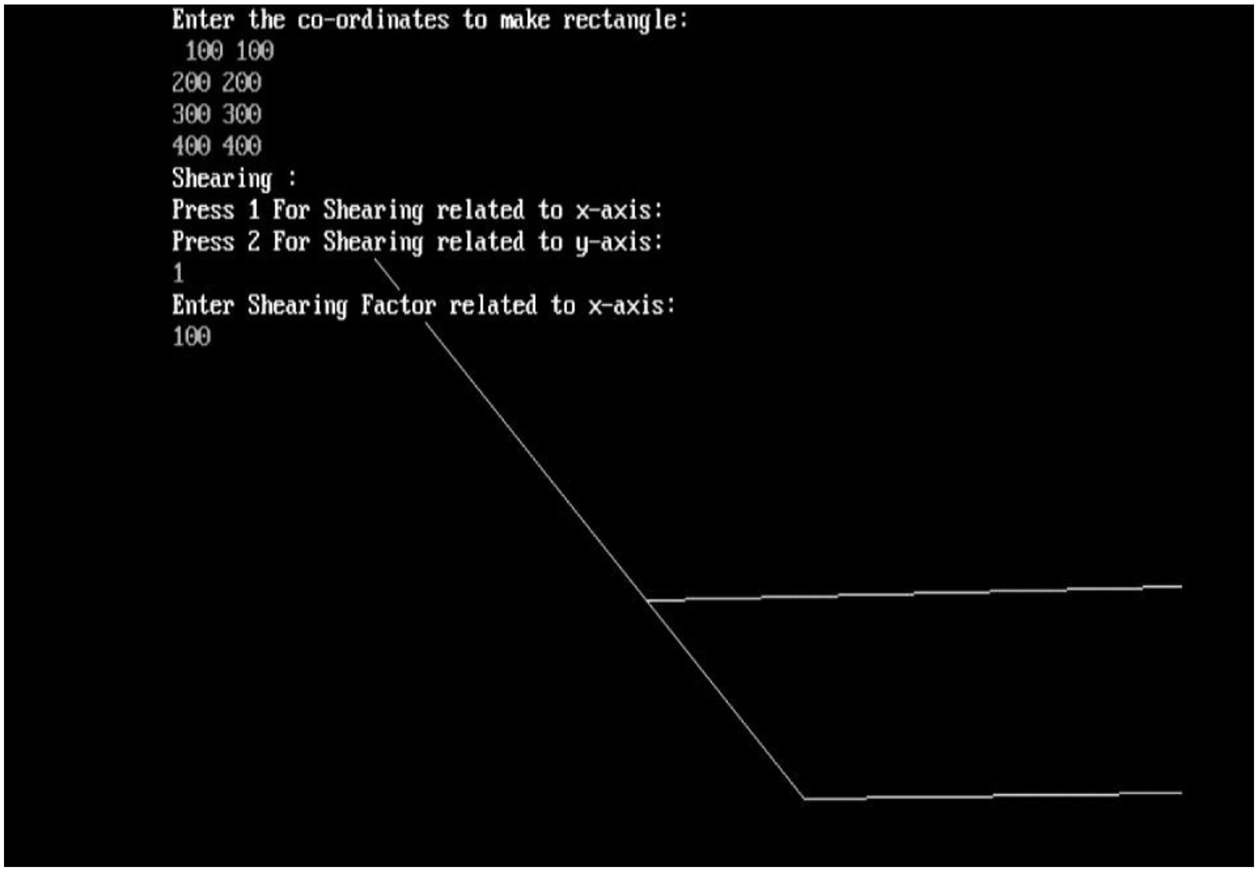
```
Enter shear factor shy along y-axis :90
Y-shear
```

15. 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");
```

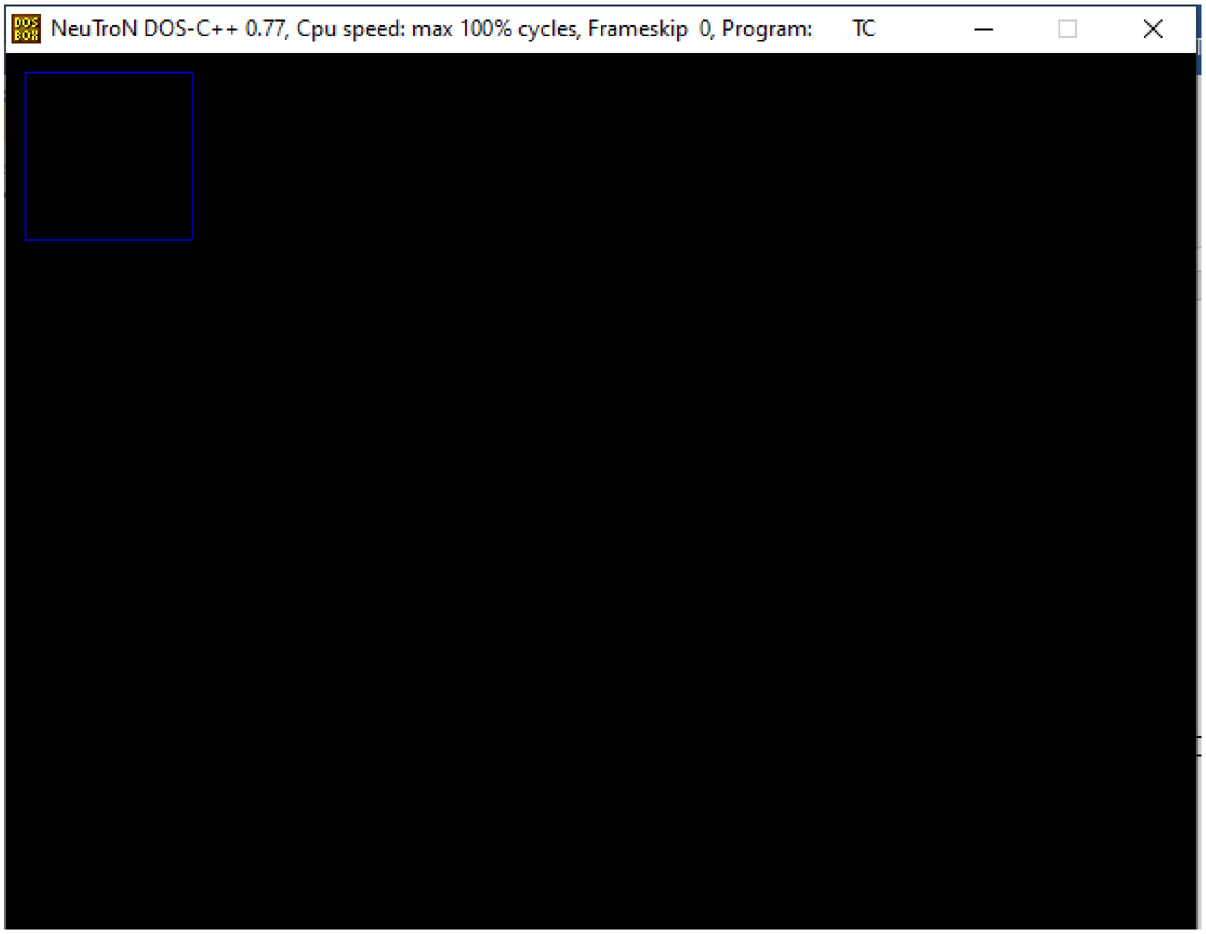
```
SCANF("%D",&SHY);
X2=X2;Y2=Y2+(X2*SHY);
X3=X3;Y3=Y3+(X3*SHY);
LINE(X,Y,X1,Y1);
LINE(X,Y,X2,Y2);
LINE(X2,Y2,X3,Y3);
LINE(X3,Y3,X1,Y1);
BREAK;
}
} GETCH();
CLOSEGRAPH();

}
```



```
Enter the co-ordinates to make rectangle:
 100 100
200 200
300 300
400 400
Shearing :
Press 1 For Shearing related to x-axis:
Press 2 For Shearing related to y-axis:
1
Enter Shearing Factor related to x-axis:
100
```

16 WAP To Make A Rectangle By Using DDA Line Algorithm .

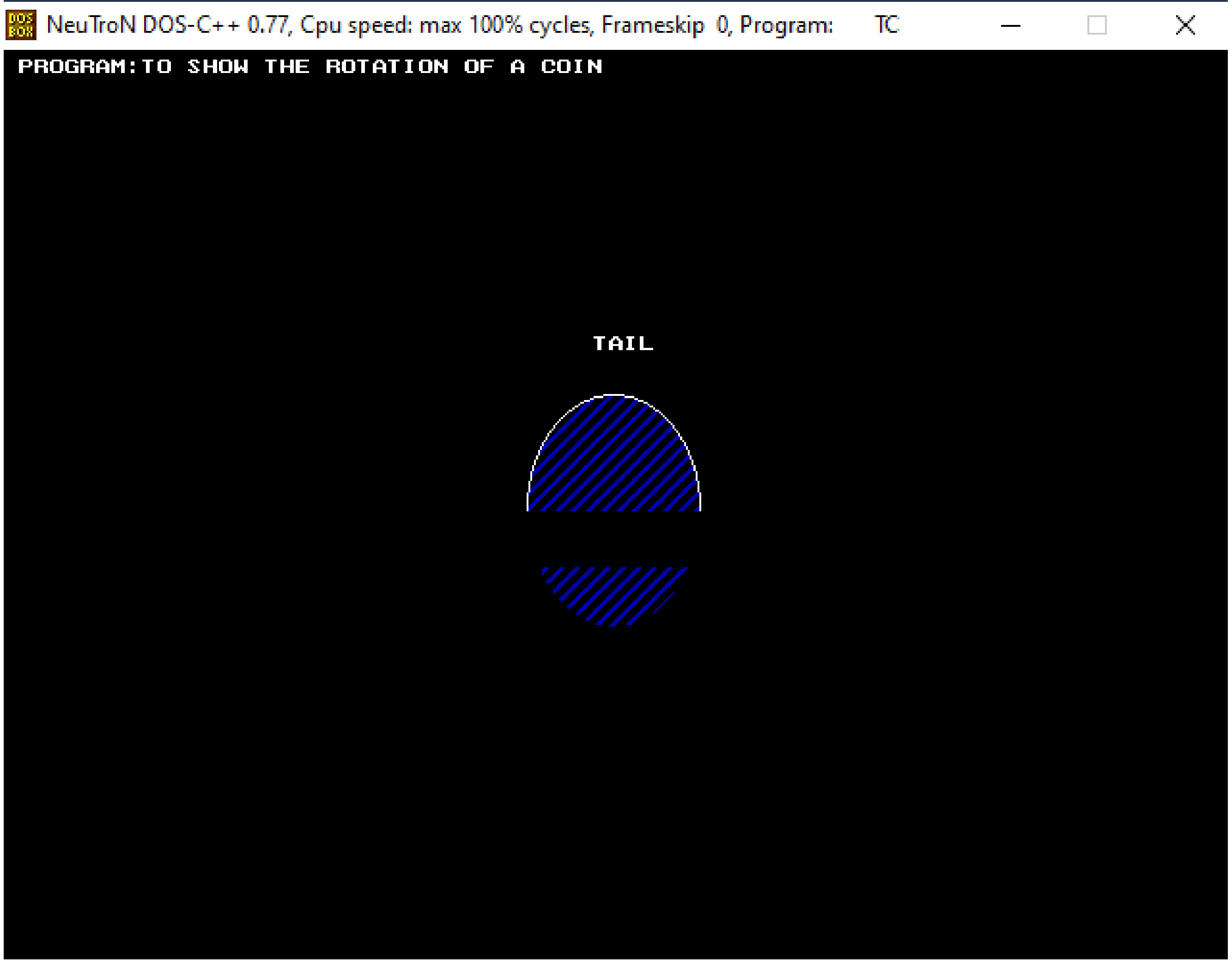
```
#INCLUDE<IOSTREAM.H>
#INCLUDE<GRAPHICS.H>
#INCLUDE<STDIO.H>
#INCLUDE<CONIO.H>
#INCLUDE<MATH.H>
VOID DDA ( INT X1, INT Y1, INT X2, INT Y2)
{
INT DX=ABS(X2-X1);INT DY=ABS(Y2-Y1);INT STEP;
IF(DX>=DY)
STEP=DX;
ELSE
STEP=DY;
IF(DX > 0)
DX=DX/STEP;
IF(DY > 0)
DY=DY/STEP;
INT X=X1;INT Y=Y1;INT I=1;
WHILE(I<=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

17 WAP To Rotate A Coin On The Table.

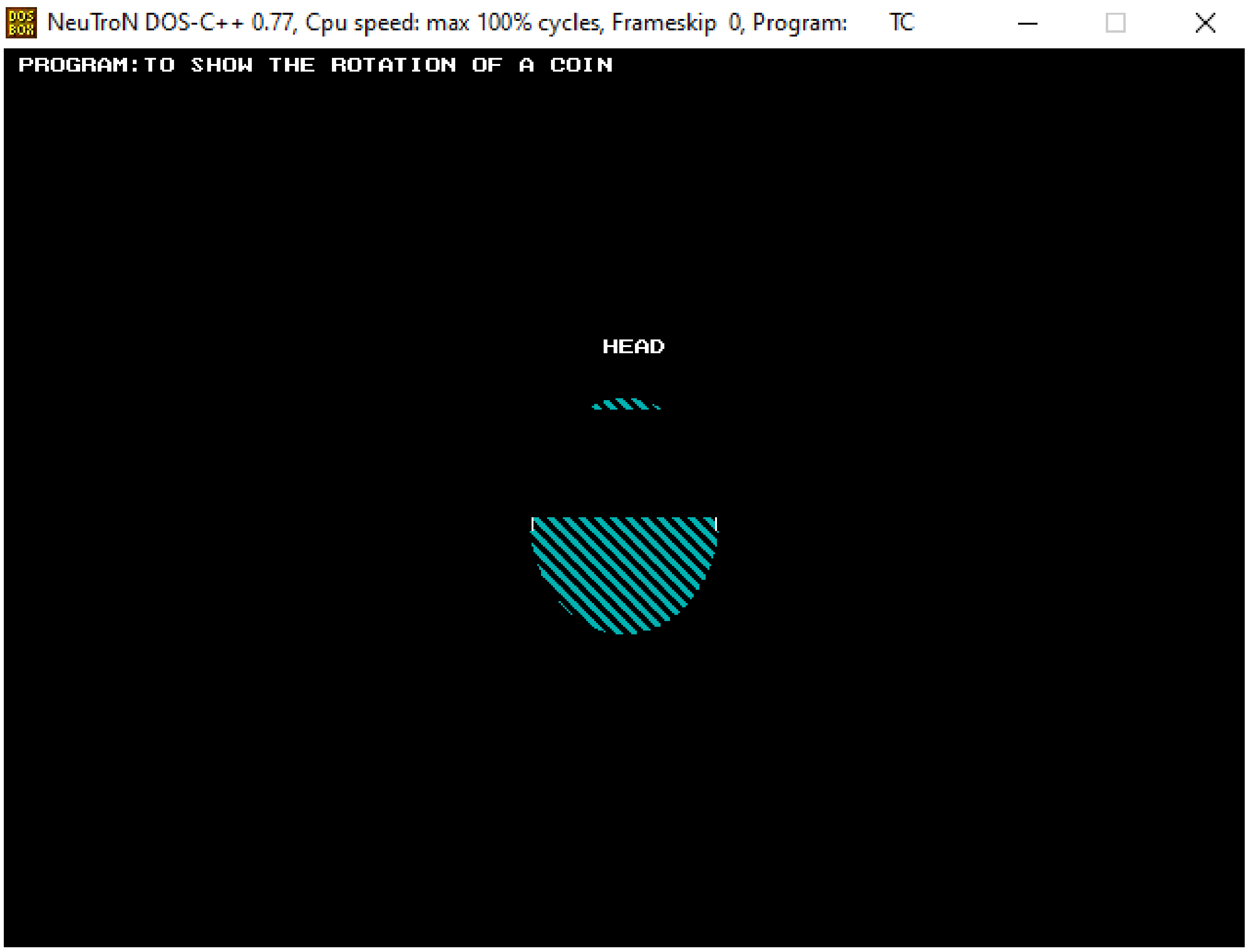
```
#INCLUDE<IOSTREAM.H>
#INCLUDE<STDIO.H>
#INCLUDE<CONIO.H>
#INCLUDE<STDLIB.H>
#INCLUDE<DOS.H>
#INCLUDE<GRAPHICS.H>VOID
MAIN(INT)
{ INT GD=DETECT,GM;
INT MIDX,MIDY,K=1,B=60,A=60;INT
XRADIUS,YRADIUS=60;
INITGRAPH(&GD,&GM,"C:\\TURBOC3\\BGI"); MIDX =
GETMAXX() / 2;MIDY = GETMAXY() / 2;
SETCOLOR(GETMAXCOLOR());
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 ACOIN");
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

TAIL
```


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

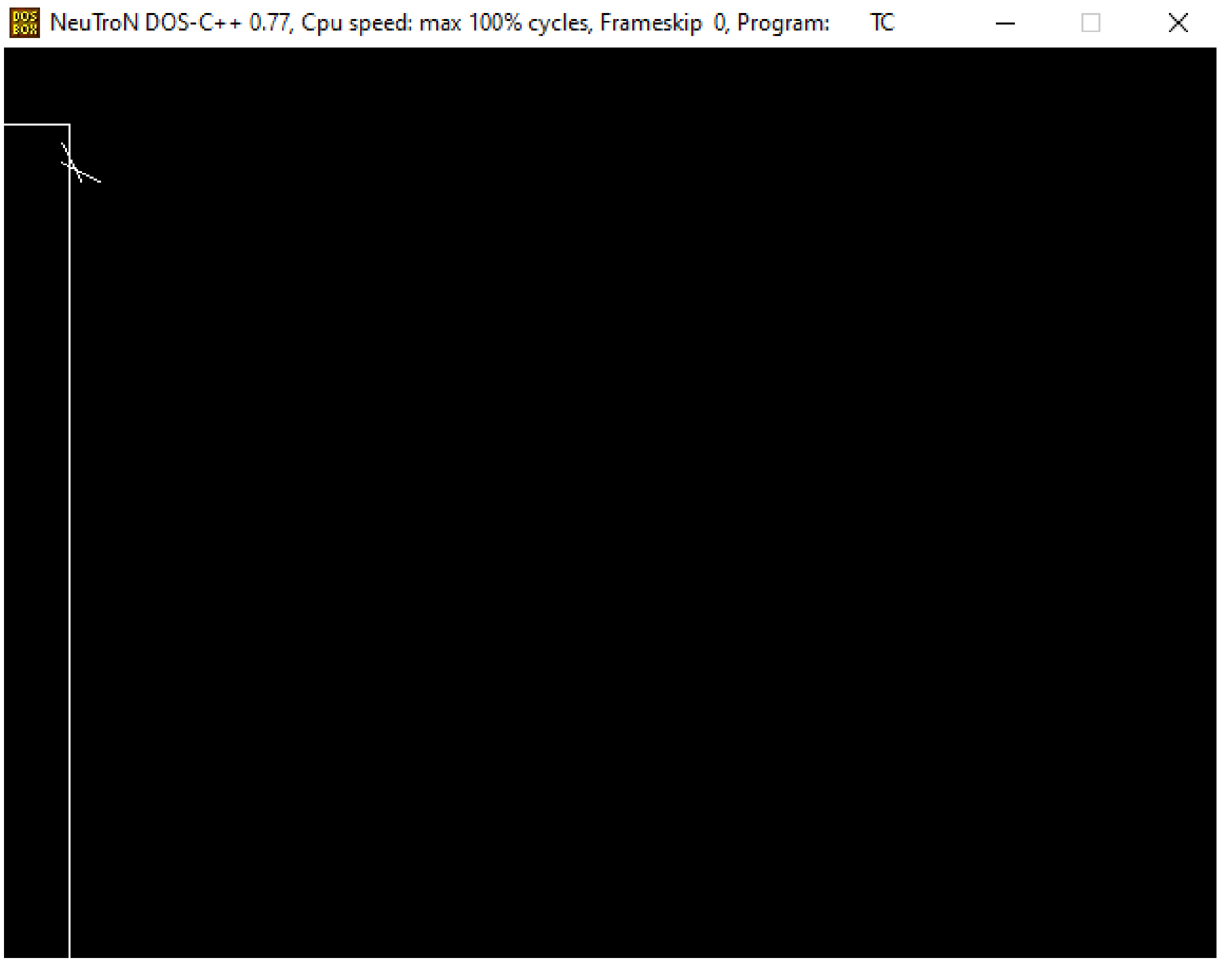
```
PROGRAM:TO SHOW THE ROTATION OF A COIN

                         HEAD
```

18 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
```

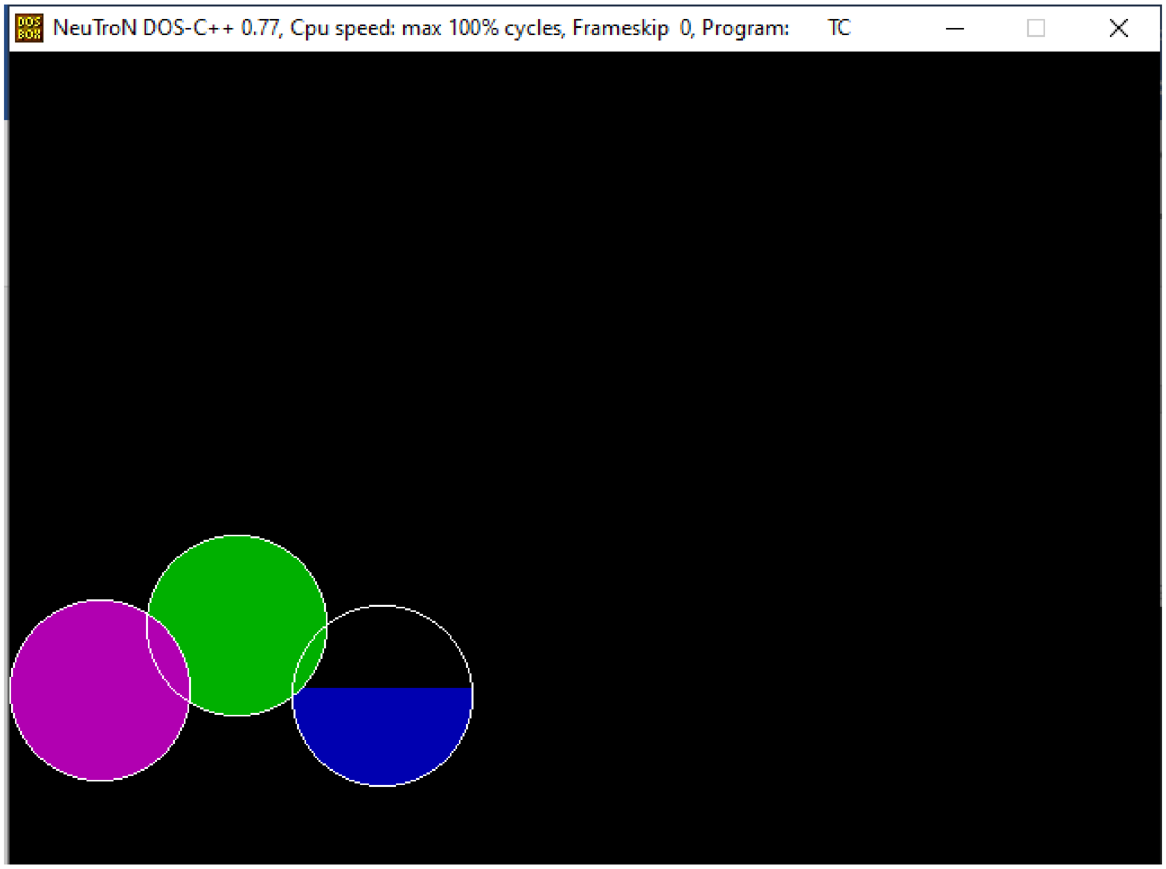
```
{ IF (OUTCODE1 == 0 && OUTCODE2 == 0)
{ ACCEPT = TRUE;
```

```
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-TOPAND
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

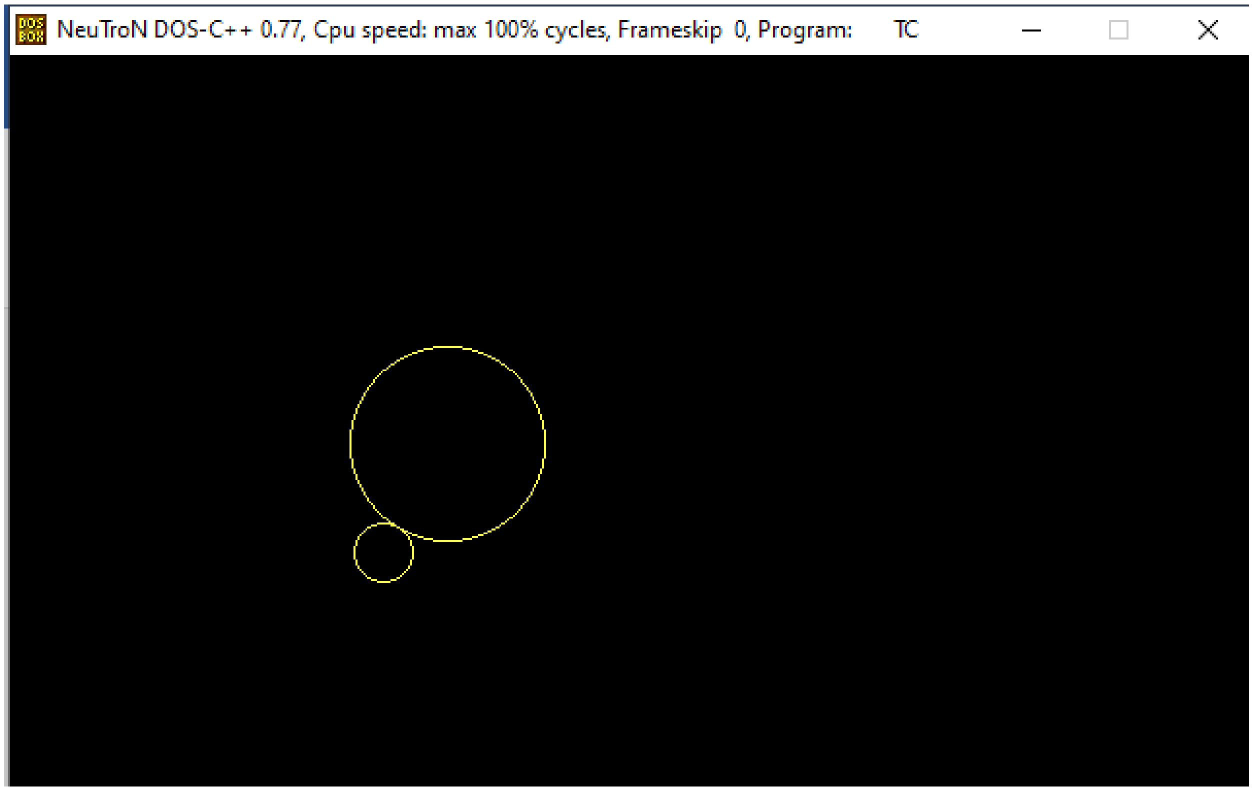
19. WAP To Make Flying Colored Balloons.

```
#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

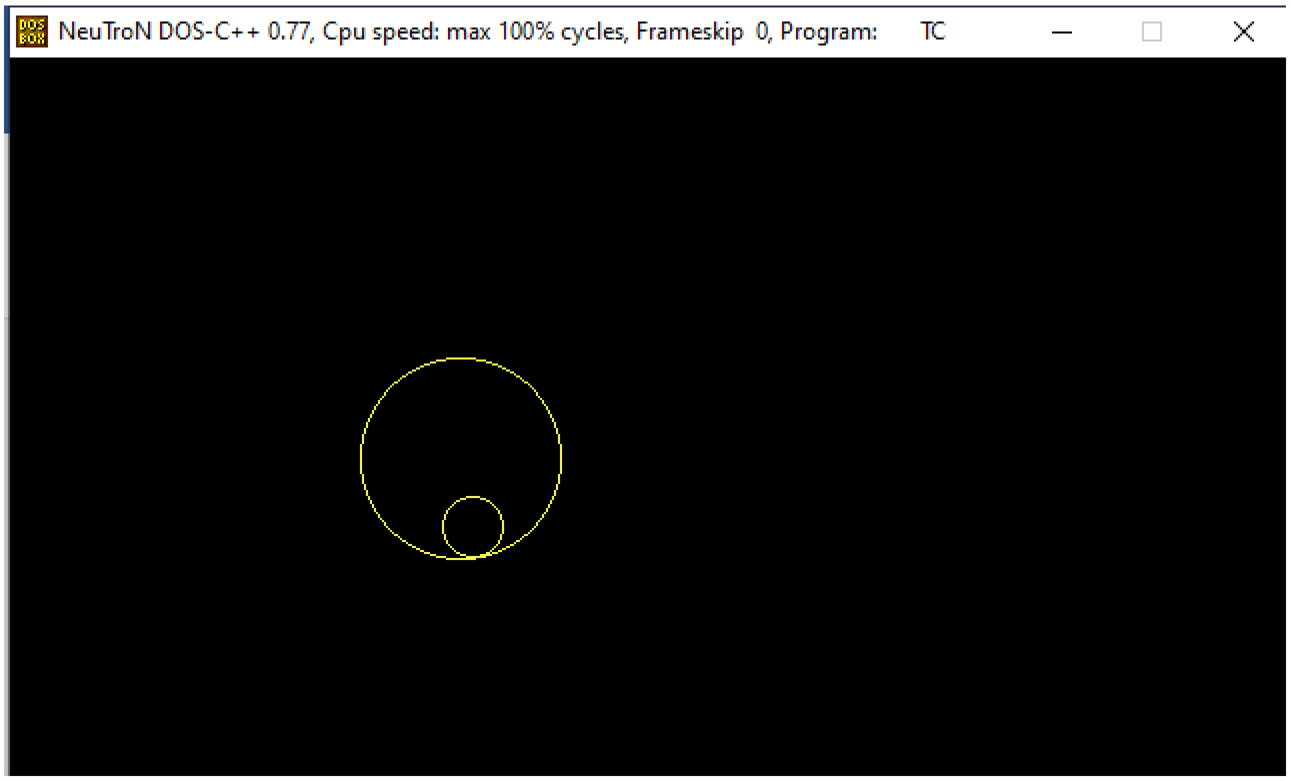
20 WAP To Rotate A Circle Outside Another Circle .

```
#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 CIRCLEA=A+1;
DELAY(50);
}
GETCH();
closegraph();
}
```


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

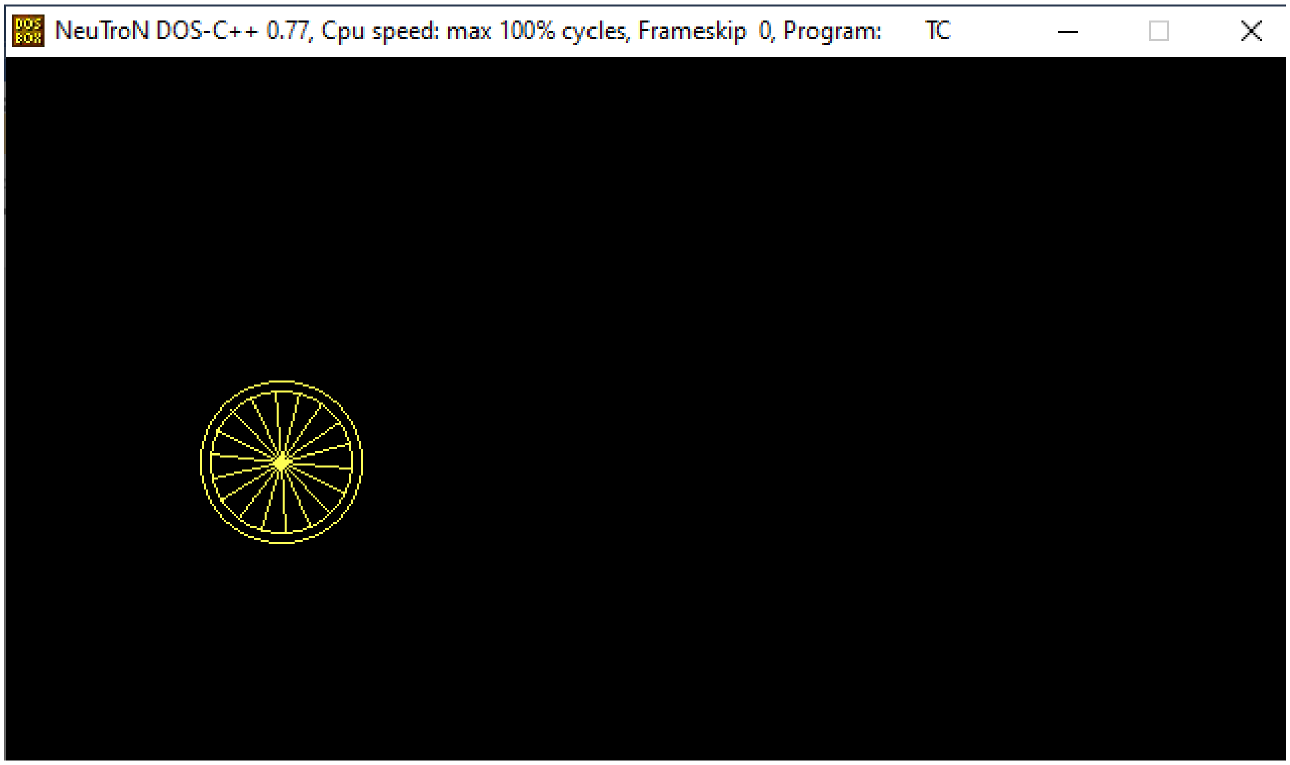
21. WAP To Rotate A Circle Inside Another Circle .

```
#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 CIRCLEA=A+1;
DELAY(50);
}GETCH();
CLOSEGRAPH();
}
```


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

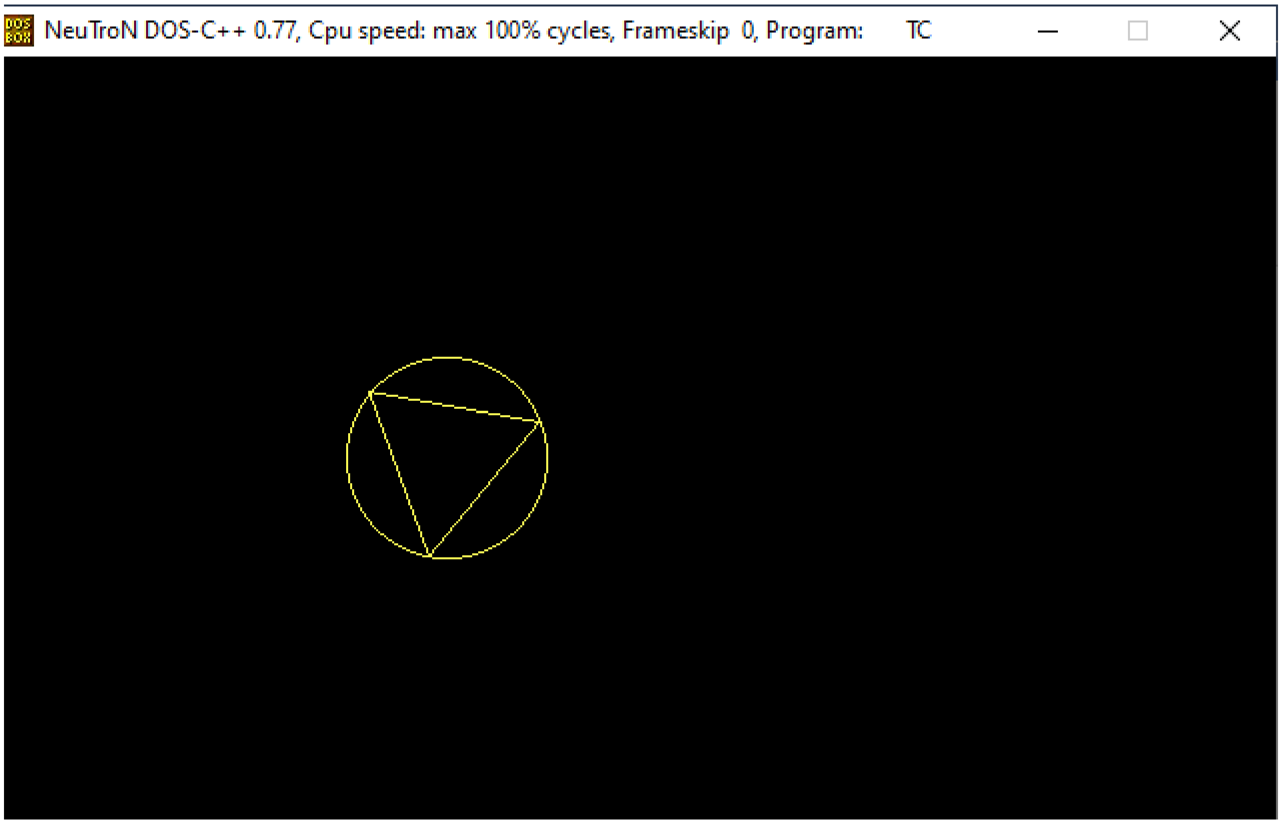
22. WAP To Make A Wheel.

```
#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();
}
```


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

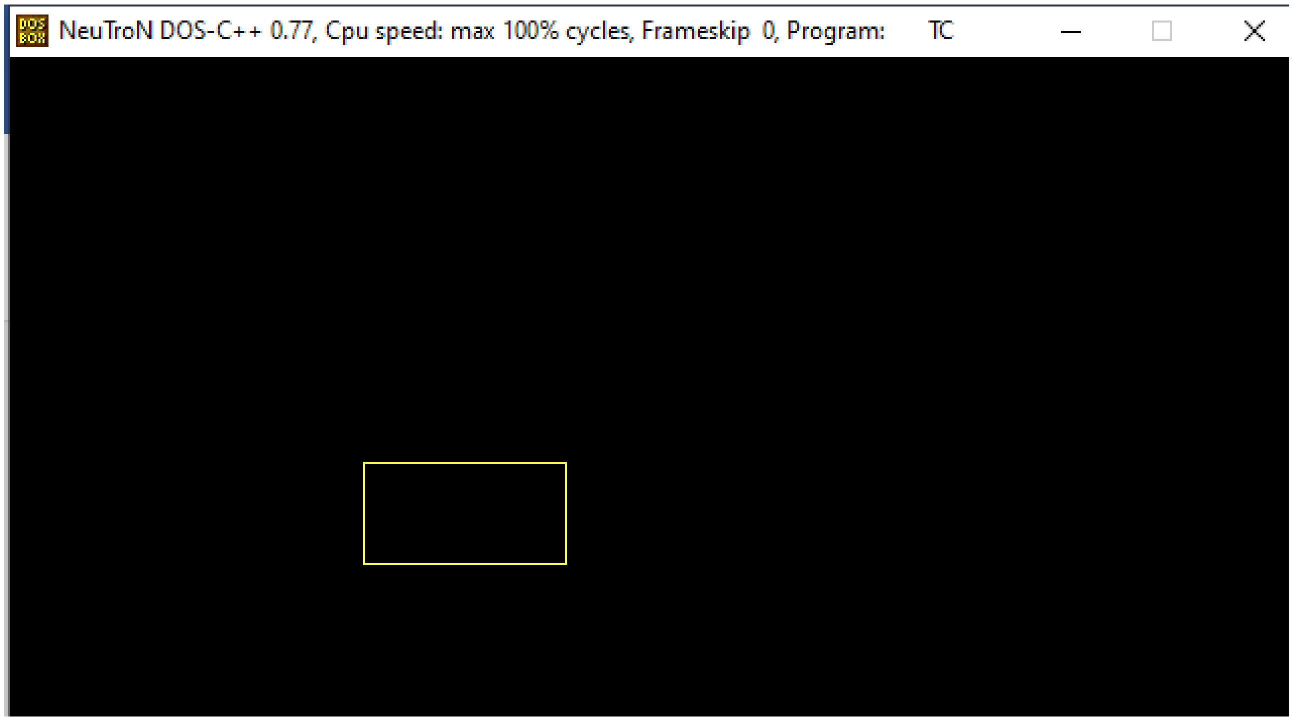
23. WAP To Rotate The Triangle At Its Centre In Clockwise Direction

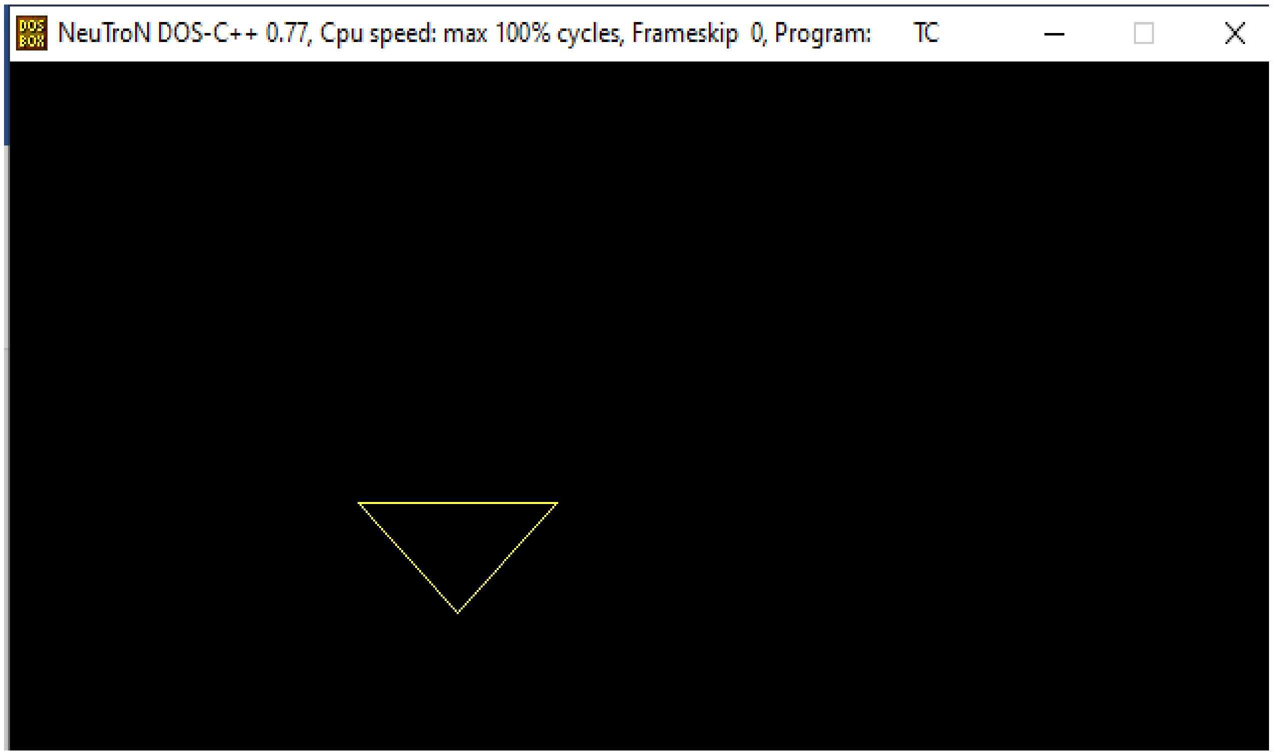
```
#INCLUDE<STDIO.H>
#INCLUDE<GRAPHICS.H>
#INCLUDE<MATH.H>
#INCLUDE<CONIO.H>
#INCLUDE<DOS.H>
INT XC=225,YC=200,R=50;INT
X[3],Y[3];
VOID DRAWCIRCLES()
{
SETCOLOR(YELLOW); CIRCLE(XC,YC,R);
LINE(X[0],Y[0],X[1],Y[1]);
LINE(X[1],Y[1],X[2],Y[2]);
LINE(X[2],Y[2],X[0],Y[0]);
}
VOID MAIN()
{
DOUBLE ANGLE=0,THETA,ANG;ANG =
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();
}
```


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

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();
}
```

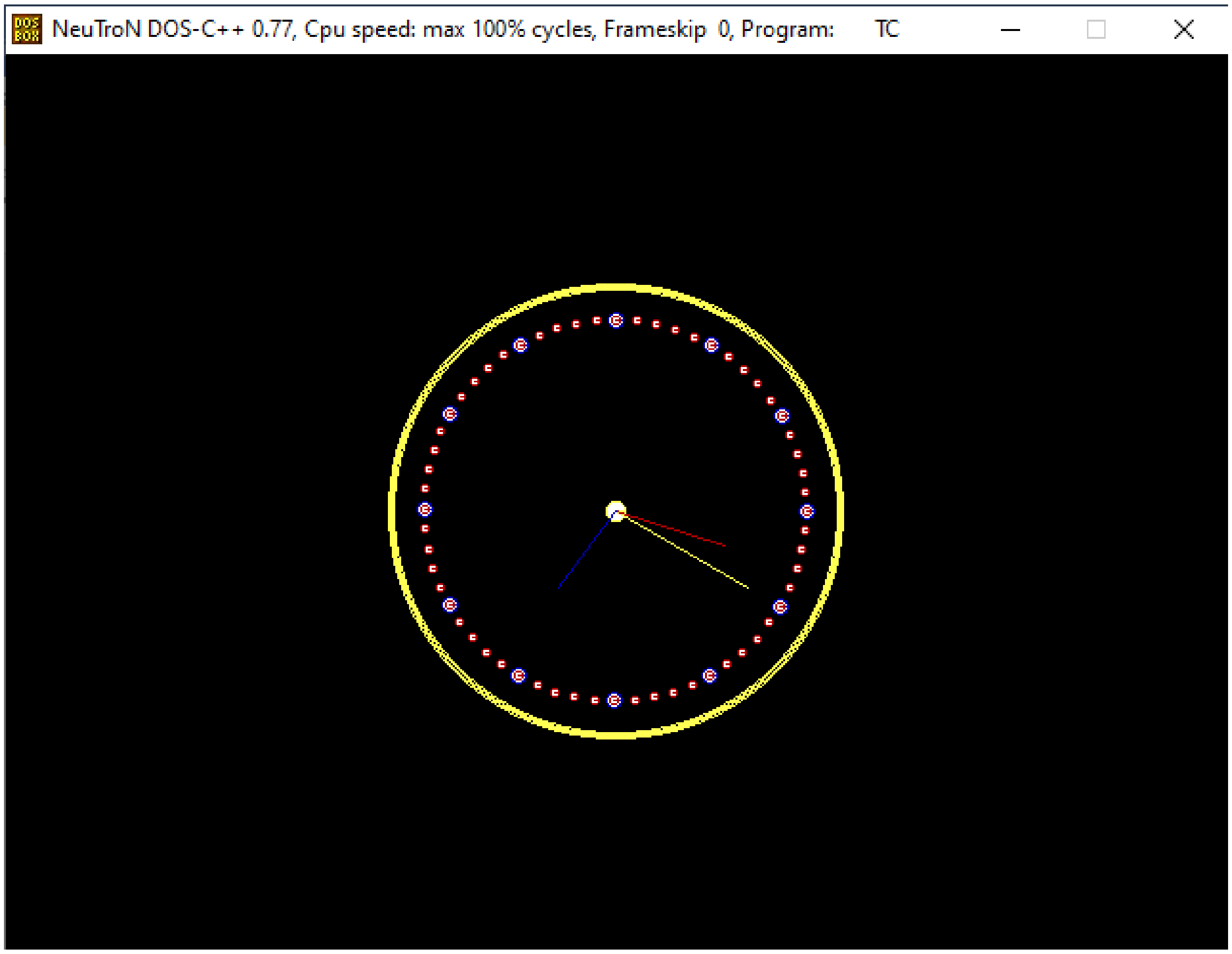

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC


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

25. 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