INDEX

BASIC GRAPHICS FUNCTION

1) INITGRAPH

- Initializes the graphics system.

Declaration

- Void far initgraph(int far *graphdriver)

Remarks

- To start the graphic system, you must first call initgraph.
- Initgraph initializes the graphic system by loading a graphics driver from disk (or validating a registered driver) then putting the system into graphics mode.
- Initgraph also resets all graphics settings (color, palette, current position, viewport, etc) totheir defaults then resets graph.

2) GETPIXEL, PUTPIXEL

- Getpixel gets the color of a specified pixel.
- Putpixel places a pixel at a specified point.

Decleration

- Unsigned far getpixel(int x, int y)

- Void far putpixel(int x, int y, int color)

Remarks

- Getpixel gets the color of the pixel located at (x,y);
- Putpixel plots a point in the color defined at (x, y).

Return value

- Getpixel returns the color of the given pixel.
- Putpixel does not return.

3) CLOSE GRAPH

- Shuts down the graphic system.

Decleration

- Void far closegraph(void);

Remarks

- Close graph deallocates all memory allocated by the graphic system.
- It then restores the screen to the mode it was in before you called initgraph.

Return value

- None.

4) ARC, CIRCLE, PIESLICE

- arc draws a circular arc.
- Circle draws a circle
- Pieslice draws and fills a circular pieslice

Declaration

- Void far arc(int x, int y, int stangle, int endangle, int radius);
- Void far circle(int x, int y, int radius);
- Void far pieslice(int x, int y, int stangle, int endangle, int radius);

Remarks

- Arc draws a circular arc in the current drawing color

- Circle draws a circle in the current drawing color
- Pieslice draws a pieslice in the current drawing color, then fills it using the current fill pattern and fill color.

5) ELLIPSE, FILL ELIPSE, SECTOR

- Ellipse draws an elliptical arc.
- Fill ellipse draws and fills ellipse.
- Sector draws and fills an elliptical pie slice.

Declaration

- Void far ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius)
- Void far fill ellipse(int x, int y, int xradius, int yradius)
- Void farsectoe(int x, int y, int stangle, int endangle, int xradius, int yradius)

Remarks

- Ellipse draws an elliptical arc in the current drawing color.
- Fill ellipse draws an elliptical arc in the current drawing color and than fills it with fill colorand fill pattern.
- Sector draws an elliptical pie slice in the current drawing color and than fills it using thepattern and color defined by setfill style or setfill pattern.

6) FLOODFILL

- Flood-fills a bounded region.

Decleration

- Void far floodfill(int x, int y, int border)

Remarks

- Floodfills an enclosed area on bitmap device.
- The area bounded by the color border is flooded with the current fill pattern and fill color.

- (x,y) is a “seed point”
    - If the seed is within an enclosed area, the inside will be filled.
    - If the seed is outside the enclosed area, the exterior will be filled.
- Use fillpoly instead of floodfill wherever possible so you can maintain code compatibility with future versions.
- Floodfill doesnot work with the IBM-8514 driver.

Return value

- If an error occurs while flooding a region, graph result returns „1‟.

7) GETCOLOR, SETCOLOR

- Getcolor returns the current drawing color.
- Setcolor returns the current drawing color.

Decleration

- Int far getcolor(void);
- Void far setcolor(int color)

Remarks

- Getcolor returns the current drawing color.
- Setcolor sets the current drawing color to color, which can range from 0 to getmaxcolor.
- To set a drawing color with setcolor , you can pass either the color number or theequivalent color name.

8) LINE,LINEREL,LINETO

- Line draws a line between two specified pints.
- Onerel draws a line relative distance from current position(CP).
- Linrto draws a line from the current position (CP) to(x,y).

Declaration

- Void far lineto(int x, int y)

Remarks

- Line draws a line from (x1, y1) to (x2, y2) using the current color, line style and thickness.It does not update the current position (CP).
- Linerel draws a line from the CP to a point that is relative distance (dx, dy) from the CP, then advances the CP by (dx, dy).
- Lineto draws a line from the CP to (x, y), then moves the CP to (x,y).

Return value

- None

9) RECTANGLE

- Draws a rectangle in graphics mode.

Decleration

- Void far rectangle(int left, int top, int right, int bottom)

Remarks

- It draws a rectangle in the current line style, thickness and drawing color.
- (left, top) is the upper left corner of the rectangle, and (right, bottom) is its lower rightcorner.

Return value

- Non

1- DDA Line Drawing Algorithm

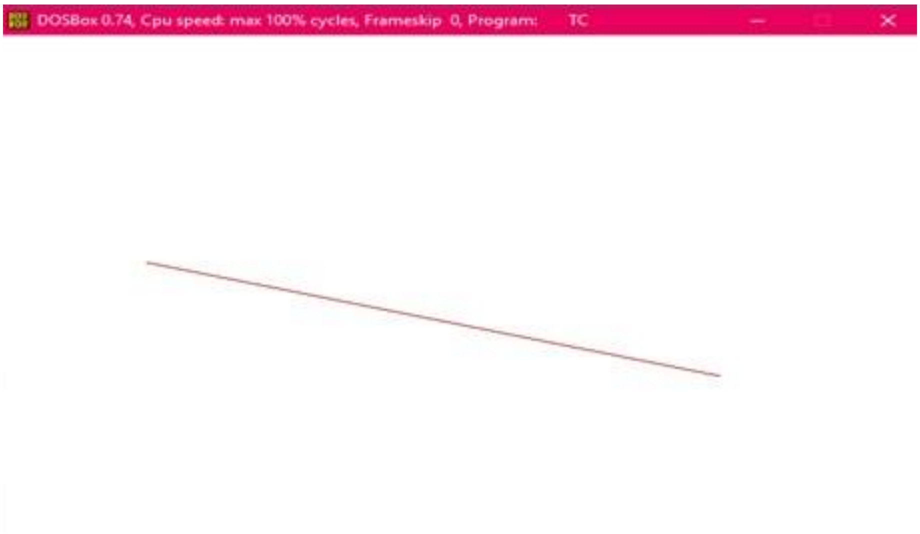
Algorithm-

```
Step1: Start Algorithm

Step2: Declare x1,y1,x2,y2,dx,dy,x,y as integer variables.

Step3: Enter value of x1,y1,x2,y2.

Step4: Calculate dx = x2-x1

Step5: Calculate dy = y2-y1

Step6: If ABS (dx) > ABS (dy)
           Then step = abs (dx)
           Else

Step7: xinc=dx/step
           yinc=dy/step
           assign x = x1
           assign y = y1

Step8: Set pixel (x, y)

Step9: x = x + xinc
           y = y + yinc
           Set pixels (Round (x), Round (y))

Step10: Repeat step 9 until x = x2

Step11: End Algorithm
```

Code-

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h> void
main()
{
    intgd = DETECT ,gm, i;
    float x, y,dx,dy,steps; int
    x0, x1, y0, y1;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    setbkcolor(WHITE);
    x0 = 100 , y0 = 200, x1 = 500, y1 = 300;
    dx = (float)(x1 - x0);
    dy = (float)(y1 - y0);
    if(dx>=dy)
           {
        steps = dx;
    }
    else
           {
        steps = dy;
    }
    dx  =  dx/steps;
    dy = dy/steps;x
    = x0;
    y = y0;i
    = 1;
    while(i<= steps)
    {
        putpixel(x, y, RED);x
        += dx;
        y += dy;
        i=i+1;
    }
    getch();
    closegraph();
}
```

OUTPUT


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

2- Bresenham’s Line Drawing Algorithm

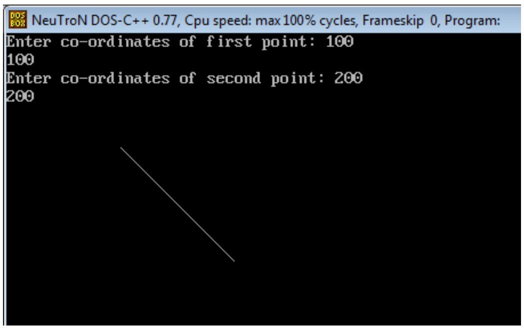
Algorithm-

```
Step1: Start Algorithm

Step2: Declare variable x1,x2,y1,y2,d,i1,i2,dx,dy

Step3: Enter value of x1,y1,x2,y2
        Where x1,y1are coordinates of starting point
        And x2,y2 are coordinates of Ending point

Step4: Calculate dx = x2-x1
        Calculate dy = y2-y1
        Calculate i1=2*dy
        Calculate i2=2*(dy-dx)
        Calculate d=i1-dx

Step5: Consider (x, y) as starting point and xendas maximum possible value of x.
        If dx < 0
            Then x = x2y
            = y2
            xend=x1
        If dx > 0
          Then x = x1y
        = y1
            xend=x2

Step6: Generate point at (x,y)coordinates.

Step7: Check if whole line is generated.
        If x > = xend
        Stop.

Step8: Calculate co-ordinates of the next pixelIf d
        < 0
          Then d = d + i1
        If d ≥ 0
     Then d = d + i2 Increment
        y = y + 1

Step9: Increment x = x + 1

Step10: Draw a point of latest (x, y) coordinates

Step11: Go to step 7

Step12: End of Algorithm
```

Code-

```
#include<stdio.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");
    printf("Enter co-ordinates of first point: ");
    scanf("%d%d", &x0, &y0);
    printf("Enter co-ordinates of second point: ");
    scanf("%d%d", &x1, &y1);
    drawline(x0, y0, x1, y1);
    return 0;  }
```

OUTPUT


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

```
Enter co-ordinates of first point: 100
100
Enter co-ordinates of second point: 200
200
```

3- Mid-Point Line Algorithm

Algorithm

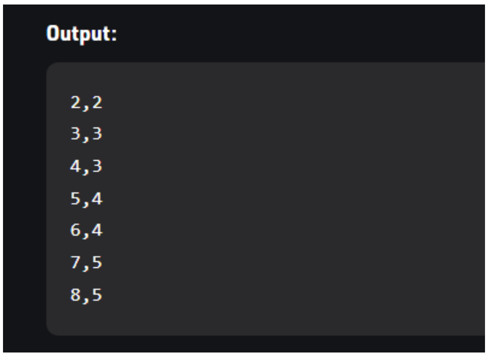
```
Input (X1,Y1) and (X2,Y2)dy
= Y2- Y1
dx = X2 - X1
// initial value of
// decision parameter d



if(dy<=dx){
d = dy - (dx/2) x
= X1 , y = Y1

// plot initial given point
Plot(x , y)

// iterate through value of X
while(x < X2)
    x = x+1

    // 'E' is chosenif
    (d < 0)
       d = d + dy

    // 'NE' is chosen
    else
       d = d + dy - dxy
       = y+1
    Plot(x,y)}
```

```
else if(dx<=dy)
{
d = dx - (dy/2) x
= X1 , y = Y1

// plot initial given point
Plot(x , y)

// iterate through value of X
while(y< Y2)
   y= y+1

   // 'E' is chosenif
   (d < 0)
      d = d + dx

   // 'NE' is chosen
   else
      d = d + dx - dy
      x= x+1
   Plot(x,y)
}
```

Code-

```
#include<bits/stdc++.h>
using namespace std;
#include<graphics.h>
void midPoint(int X1, int Y1, int X2, int Y2)
{
        int dx = X2 - X1;
        int dy = Y2 - Y1;
        if(dy<=dx){
        int d = dy - (dx/2);int
        x = X1, y = Y1;
        cout << x << "," << y << "\n";
        while (x < X2)
        {
                x++;
                if (d < 0)
                        d = d + dy;
                else
                {

                        d += (dy - dx);
                        y++;
                }

                cout << x << "," << y << "\n";
        }
        }

else if(dx<dy)
{
        int d = dx - (dy/2);int
        x = X1, y = Y1;
        cout << x << "," << y << "\n";
        while (y < Y2)
        {
                y++;
                if (d < 0)
                        d = d + dx;
                        else
                        {
                                    d += (dx - dy);
                                    x++;
                        }
                        cout << x << "," << y << "\n";
            }
}
}
int main()
{           int gd = DETECT, gm;
            initgraph (&gd, &gm, "");

            int X1 = 2, Y1 = 2, X2 = 8, Y2 = 5;
            midPoint(X1, Y1, X2, Y2);
            return 0;
}
```

OUTPUT



Output:

```
2,2
3,3
4,3
5,4
6,4
7,5
8,5
```

4- Bresenham’s Circle Algorithm

Algorithm

Step1: Start Algorithm

Step2: Declare p, q, x, y, r, d variables

p, q are coordinates of the center of the circler is

the radius of the circle

Step3: Enter the value of r

Step4: Calculate d = 3 - 2r

Step5: Initialize x=0

&nbsy= r

Step6: Check if the whole circle is scan convertedIf x

> = y

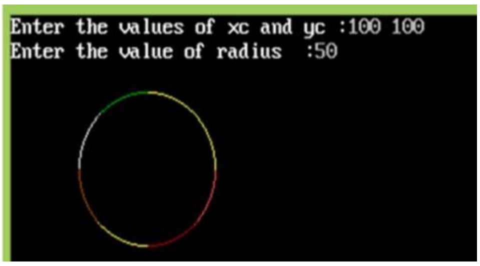
Stop

Code

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

  void    EightWaySymmetricPlot(int xc,int yc,int x,int y)
 {
  putpixel(x+xc,y+yc,RED);
  putpixel(x+xc,-y+yc,YELLOW);
  putpixel(-x+xc,-y+yc,GREEN);
  putpixel(-x+xc,y+yc,YELLOW);
  putpixel(y+xc,x+yc,12);
  putpixel(y+xc,-x+yc,14);
  putpixel(-y+xc,-x+yc,15);
  putpixel(-y+xc,x+yc,6);
 }

  void BresenhamCircle(int xc,int yc,int r)
 {
  int x=0,y=r,d=3-(2*r);
  EightWaySymmetricPlot(xc,yc,x,y);

  while(x<=y)
   {
    if(d<=0)
           {
      d=d+(4*x)+6;
    }
   else
    {
      d=d+(4*x)-(4*y)+10;
      y=y-1;
    }
     x=x+1; EightWaySymmetricPlot(xc,yc,x,y);
    }
  }

  int  main(void)
 {
  /* request auto detection */
```

```
int xc,yc,r,gdriver = DETECT, gmode, errorcode;
/* initialize graphics and local variables */
 initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");

 /* read result of initialization */
 errorcode = graphresult();

  if (errorcode != grOk)  /* an error occurred */
 {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1); /* terminate with an error code */
 }
   printf("Enter the values of xc and yc :");
   scanf("%d%d",&xc,&yc);
   printf("Enter the value of radius :");
   scanf("%d",&r); BresenhamCircle(xc,yc,r);

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

OUTPUT



```
Enter the values of xc and yc :100 100
Enter the value of radius  :50
```

5- Translation in 2D

Algorithm

X' = Dx + XY'

= Dy + Y

or P' = T + P where

P' = (X', Y'), T

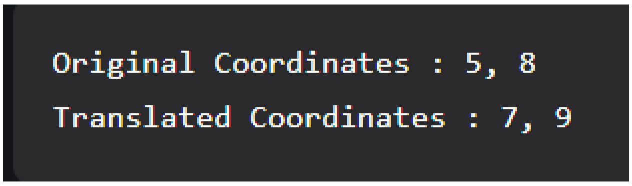
= (Dx, Dy),

P = (X, Y)

Code

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void translatePoint ( int P[], int T[])
{
        int gd = DETECT, gm, errorcode;
        initgraph (&gd, &gm, "c:\\tc\\bgi");
        cout<<"Original Coordinates :"<<P[0]<<","<<P[1];
        putpixel (P[0], P[1], 1);
        P[0] = P[0] + T[0];
        P[1] = P[1] + T[1];
        cout<<"\nTranslated Coordinates :"<< P[0]<<","<< P[1];
        putpixel (P[0], P[1], 3);
        closegraph();
}
int main()
{
        int P[2] = {5, 8}; // coordinates of point
        int T[] = {2, 1}; // translation factor

        translatePoint (P, T);

        return 0;

}
```

OUTPUT



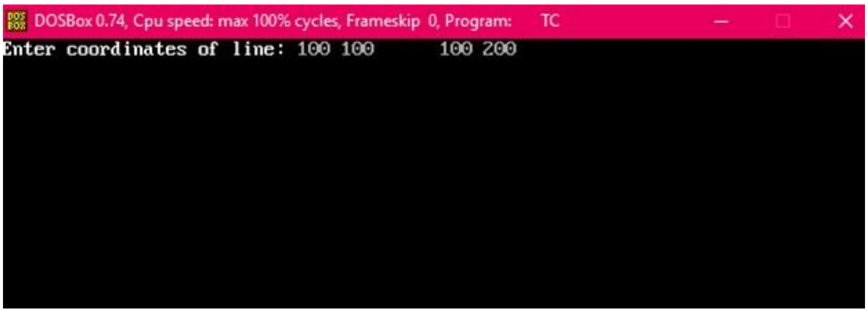
```
Original Coordinates : 5, 8
Translated Coordinates : 7, 9
```

6- Rotation in 2D

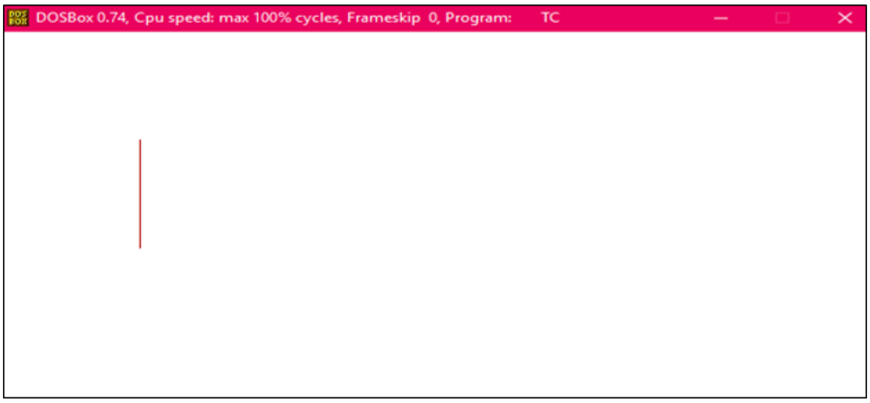
Code

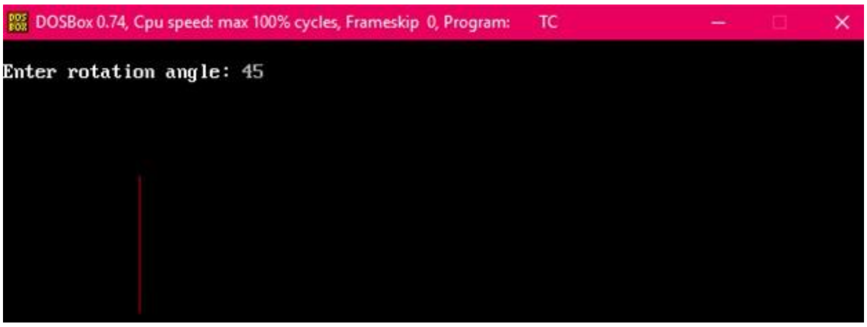
```
#include<stdio.h>

#include<graphics.h>

#include<math.h>

int main()

{

    intgd=0,gm,x1,y1,x2,y2;

    double s,c, angle;

    initgraph(&gd, &gm, "C:\\TC\\BGI");

    setcolor(RED);

    printf("Enter coordinates of line: ");

    scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

    cleardevice();

    setbkcolor(WHITE);

    line(x1,y1,x2,y2);

    getch();

    setbkcolor(BLACK);

    printf("Enter rotation angle: ");

    scanf("%lf", &angle);

    setbkcolor(WHITE);

    c = cos(angle *3.14/180); s

    = sin(angle *3.14/180); x1 =

    floor(x1 * c + y1 * s);

    y1 = floor(-x1 * s + y1 * c);x2

    = floor(x2 * c + y2 * s); y2 =

    floor(-x2 * s + y2 * c);

    cleardevice();

    line(x1, y1 ,x2, y2);

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

OUTPUT

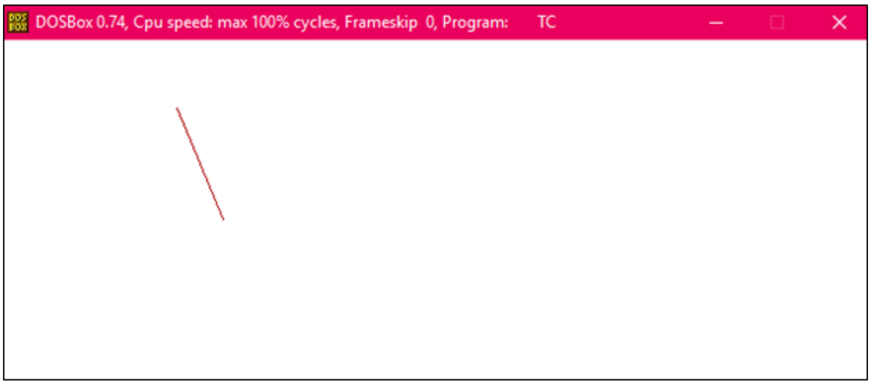

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter coordinates of line: 100 100       100 200
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
Enter rotation angle: 45
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

7- Scaling in 2D

Algorithm-

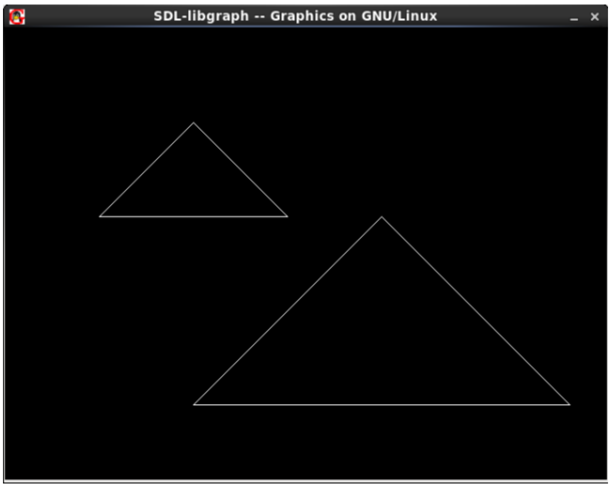
```
1.Make a 2x2 scaling matrix S as:
   Sx 0
   0 Sy
2. For each point of the polygon.
   (i)Make a 2x1 matrix P, where P[0][0] equals
      to x coordinate of the point and P[1][0]
      equals to y coordinate of the point.
   (ii)Multiply scaling matrix S with point
       matrix P to get the new coordinate.
3. Draw the polygon using new coordinates.
```

Code-

```
#include<stdio.h>
#include<graphics.h>
void findNewCoordinate(int s[][2], int p[][1])
{
	int temp[2][1] = { 0 };

	for (int i = 0; i < 2; i++)
		for (int j = 0; j < 1; j++)
			for (int k = 0; k < 2; k++)
				temp[i][j] += (s[i][k] * p[k][j]);
	p[0][0] = temp[0][0];
	p[1][0] = temp[1][0];
}
void scale(int x[], int y[], int sx, int sy)
{
	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]);
        int s[2][2] = { sx, 0, 0, sy };int
        p[2][1];
        for (int i = 0; i < 3; i++)
        {
                p[0][0] = x[i];
                p[1][0] = y[i];

                findNewCoordinate(s, p);

                x[i] = p[0][0];
                y[i] = p[1][0];
        }
        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]);
}
int main()
{
        int x[] = { 100, 200, 300 };
        int y[] = { 200, 100, 200 };
        int sx = 2, sy = 2;

        int gd, gm;
        detectgraph(&gd, &gm);
        initgraph(&gd, &gm," ");

        scale(x, y, sx,sy);
        getch();
        return 0;
}
```

OUTPUT



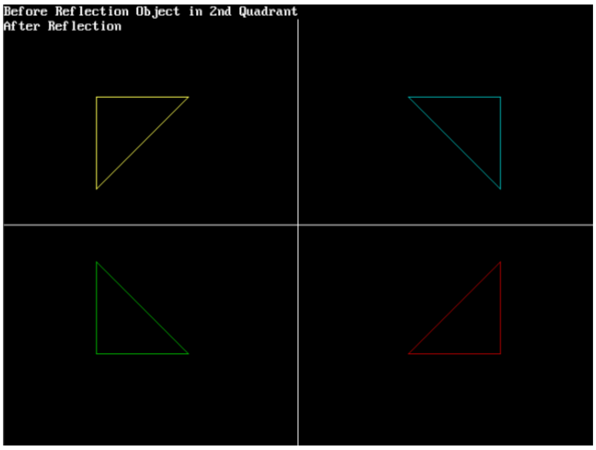
8- Reflection in 2D

Code

```
#include <conio.h>

#include <graphics.h>

#include <stdio.h>

Code

void main()

{

        int gm, gd = DETECT, ax, x1 = 100;int

        x2 = 100, x3 = 200, y1 = 100;

        int y2 = 200, y3 = 100;

        initgraph(&gd, &gm, "");

        cleardevice();

        line(getmaxx() / 2, 0, getmaxx() / 2,

                getmaxy());

        line(0, getmaxy() / 2, getmaxx(),

                getmaxy() / 2);

        printf("Before Reflection Object""

                in 2nd Quadrant");

        setcolor(14); line(x1,

        y1, x2, y2);

        line(x2, y2, x3, y3);

        line(x3, y3, x1, y1);

        getch();

        printf("\nAfter Reflection");

        setcolor(4);

        line(getmaxx() - x1, getmaxy() - y1,

                getmaxx() - x2, getmaxy() - y2);

        line(getmaxx() - x2, getmaxy() - y2,

                getmaxx() - x3, getmaxy() - y3);
```

```
    line(getmaxx() - x3, getmaxy() - y3,
            getmaxx() - x1, getmaxy() - y1);
    setcolor(3); line(getmaxx() -
    x1, y1,
            getmaxx() - x2, y2);
    line(getmaxx() - x2, y2,
            getmaxx() - x3, y3);
    line(getmaxx() - x3, y3,
            getmaxx() - x1, y1);
    setcolor(2);
    line(x1, getmaxy() - y1, x2,
            getmaxy() - y2);
    line(x2, getmaxy() - y2, x3,
            getmaxy() - y3);
    line(x3, getmaxy() - y3, x1,
            getmaxy() - y1);
    getch();
    closegraph();
}
```

OUTPUT



```
Before Reflection Object in 2nd Quadrant
After Reflection
```

9- Shearing in 2D

Algorithm

Given,

Old corner coordinates of the triangle = A (1, 1), B(0, 0), C(1, 0)

Shearing parameter along X-axis (Shx) = 4

Shearing parameter along Y-axis (Shy) = 1

Along x-axis:

A'=(1+4*1, 1)=(5, 1)

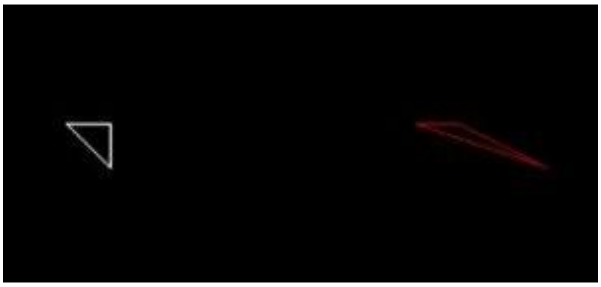
B'=(0+4*0, 0)=(0, 0)

C'=(1+4*0, 0)=(1, 0)

Code

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h> void
main()
{
int gd=DETECT,gm;
int x,y,x1,y1,x2,y2,shear_f;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("\n please enter first coordinate = ");
scanf("%d %d",&x,&y);
printf("\n please enter second coordinate = ");
scanf("%d %d",&x1,&y1);
printf("\n please enter third coordinate = ");
scanf("%d %d",&x2,&y2);
printf("\n please enter shearing factor x = ");
scanf("%d",&shear_f);
cleardevice();
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
setcolor(RED); x=x+
y*shear_f; x1=x1+
y1*shear_f;x2=x2+
y2*shear_f;
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
getch();
closegraph();
}
```

OUTPUT



10- Cohen’s Sutherland Algorithm

Algorithm

1. Read 2 end points of line as p1(x1,y1) and p2(x2,y2

2. Read 2 corner points of the clipping window (left-top and right-bottom) as (wx1,wy1) and (wx2,wy2)

3. Assign the region codes for 2 endpoints p1 and p2 using following steps:-

initialize code with 0000

Set bit 1 if x<wx1

Set bit 2 if x>wx2

Set bit 3 if y<wy2

Set bit 4 if y>wy1

4. Check for visibility of line

If region codes for both endpoints are zero then line is completely visible. Draw the line go tostep 9.

If region codes for endpoints are not zero and logical ANDing of them is also nonzero thenline is invisible. Discard the line and move to step 9.

If it does not satisfy 4.a and 4.b then line is partially visible.

5. Determine the intersecting edge of clipping window as follows:-

If region codes for both endpoints are nonzero find intersection points p1’ and p2’ with boundary edges.

If region codes for any one end point is non zero then find intersection point p1’ or p2’.

6. Divide the line segments considering intersection points.

7. Reject line segment if any end point of line appears outside of any boundary.

8. Draw the clipped line segment.

9. Stop.

Code

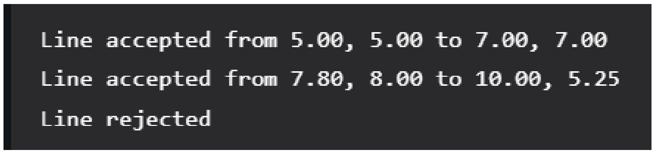
```
#include <iostream>using

namespace std;

const int INSIDE = 0; // 0000

const int LEFT = 1; // 0001

const int RIGHT = 2; // 0010
const int BOTTOM = 4; // 0100
const int TOP = 8; // 1000 const
int x_max = 10;
const  int  y_max  =  8;
const  int  x_min  =  4;
const int y_min = 4;
int computeCode(double x, double y)
{
        // initialized as being insideint
        code = INSIDE;

        if (x < x_min) // to the left of rectangle
                code |= LEFT;
        else if (x > x_max) // to the right of rectangle
                code |= RIGHT;
        if (y < y_min) // below the rectangle
                code |= BOTTOM;
        else if (y > y_max) // above the rectangle
                code |= TOP;

        return code;
}
void cohenSutherlandClip(double x1, double y1,
                                        double x2, double y2)
{
        int code1 = computeCode(x1, y1);int
        code2  =  computeCode(x2,  y2); bool
        accept = false;

        while (true) {
```

```
if ((code1 == 0) && (code2 == 0)) {
        accept = true;
        break;
}
else if (code1 & code2) {
        break;
}
else {
        int code_out;
        double x, y;
        if (code1 != 0)
                code_out = code1;
        else
                code_out = code2;
        if (code_out & TOP) {
                x = x1 + (x2 - x1) * (y_max - y1) / (y2 - y1);y =
                y_max;
        }
        else if (code_out & BOTTOM) {
                x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1);y =
                y_min;
        }
        else if (code_out & RIGHT) {
                y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1);x =
                x_max;
        }
        else if (code_out & LEFT) {
                y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1);x =
                x_min;
        }
```

```
            if (code_out == code1) {
                x1 = x;
                y1 = y;
                code1 = computeCode(x1, y1);
            }
            else {

                x2 = x;
                y2 = y;
                code2 = computeCode(x2, y2);
            }
        }
    }
    if (accept) {
        cout << "Line accepted from " << x1 << ", "
                 << y1 << " to " << x2 << ", " << y2 << endl;
    }
    else

        cout << "Line rejected" << endl;
}
```

```
int main()
{
        cohenSutherlandClip(5, 5, 7, 7);
        cohenSutherlandClip(7, 9, 11, 4);
        cohenSutherlandClip(1, 5, 4, 1);

        return 0;
}
```

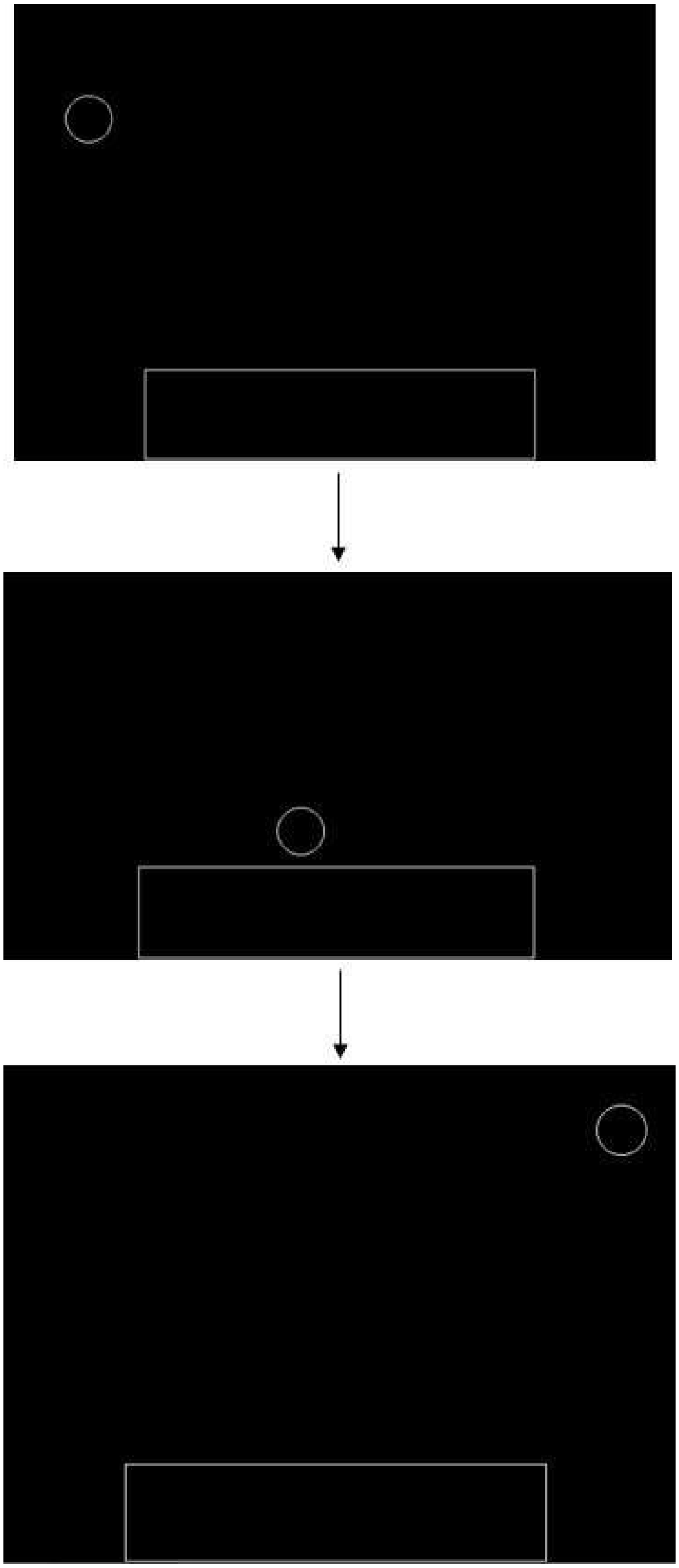
OUTPUT



```
Line accepted from 5.00, 5.00 to 7.00, 7.00
Line accepted from 7.80, 8.00 to 10.00, 5.25
Line rejected
```

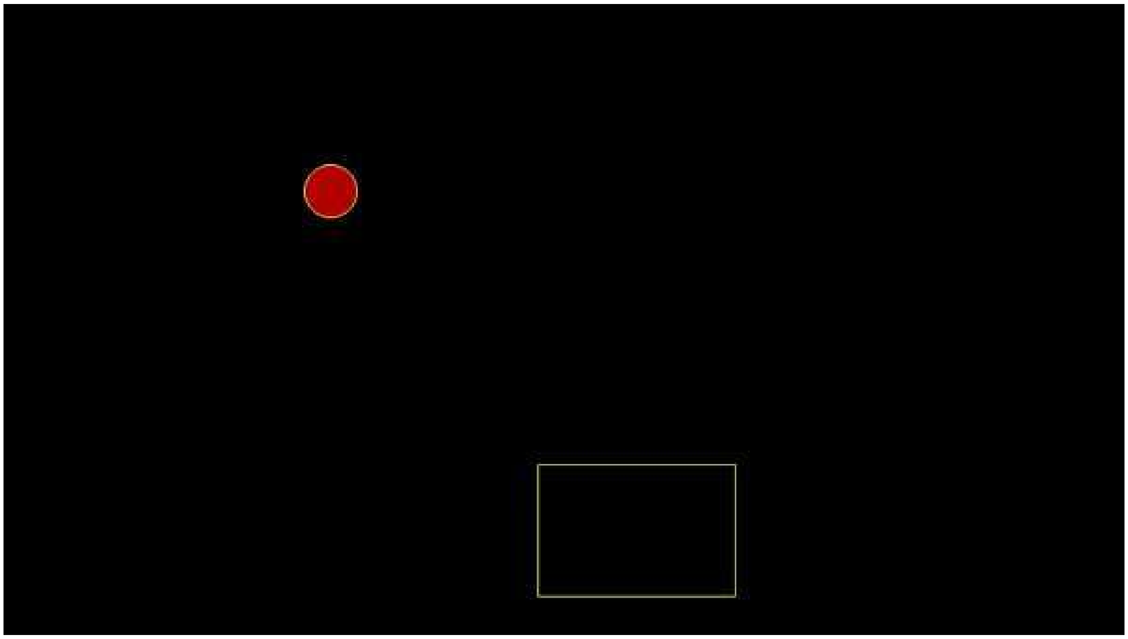
11. Program to create bouncing ball

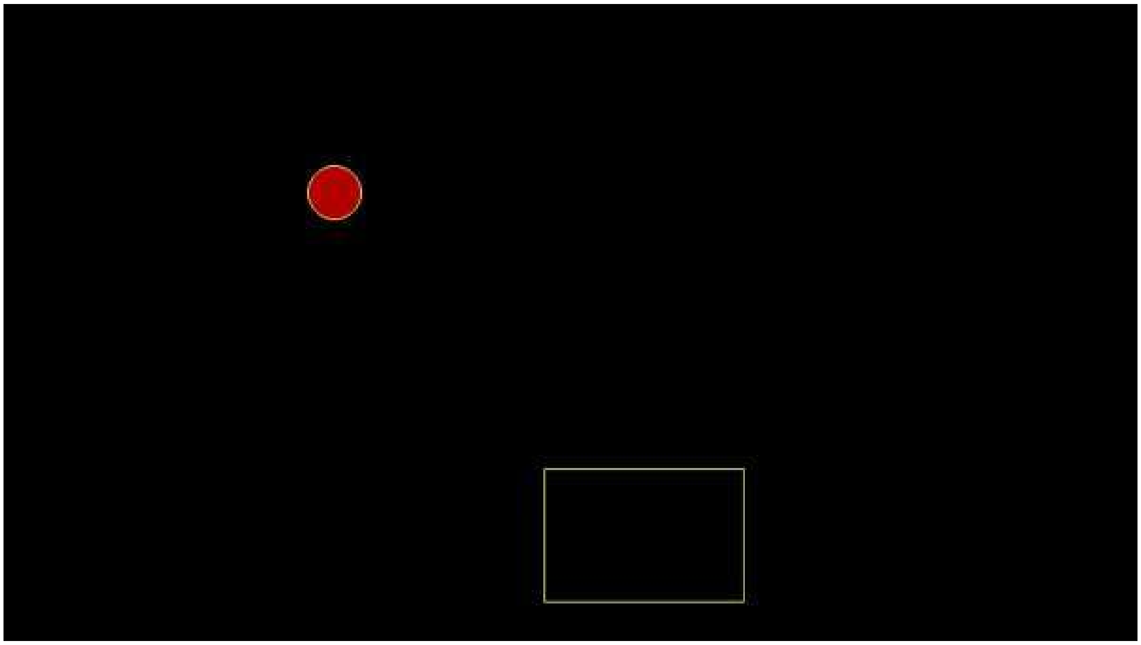
```
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
  clrscr();
  int  gd=DETECT,gm,x1=150,y1=400,x2=489,y2=478,i,j,k,l;
   initgraph(&gd,&gm,"c:\\turboc3\\bgi");

  for(i=50,j=130;i<300,j<380;i++,j++)
  {  rectangle(x1,y1,x2,y2);
   circle(i,j,20);
  delay(10); clearviewport();
    }
   for(k=300,l=380;k<500,l>130;k++,l--
   )
   {
   rectangle(x1,y1,x2,y2);
  circle(k,l,20);
  delay(10);
  clearviewport(); }
  getch();
}
```

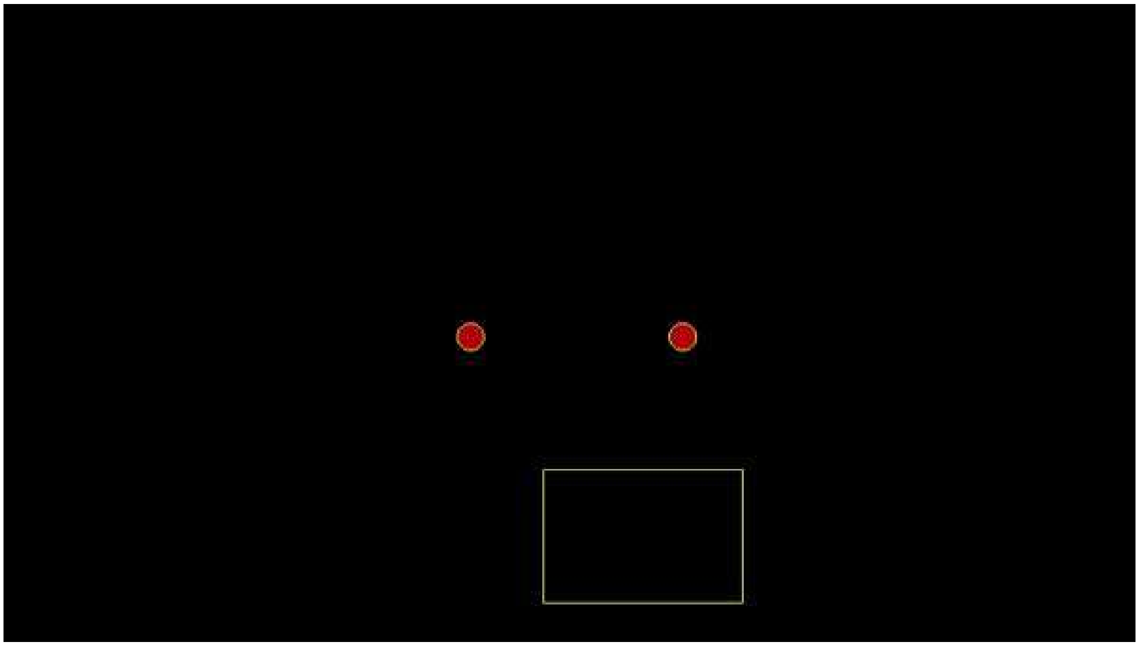


12. Program to create splitting ball into two parts

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>  void
main()
{
    Intgd=DETECT,gm,x1=30,y1=30,rad=20,i,
   j;
    initgraph(&gd,&gm,"c:\\turboc3\\bgi");
   setcolor(YELLOW);
for(i=0,j=0; i<95,j<300; i++,j++)
   {
rectangle(300,350,450,450);
setfillstyle(SOLID_FILL,RED);
fillellipse(x1+i,y1+j,rad,rad);
delay(10);
   cleardevice();
}
      x1=325; y1=330; for(i=0,j=0;
      i<200,j<200; i++,j++)

{ rectangle(300,350,450,450);
setfillstyle(SOLID_FILL,RED);
fillellipse(x1-i,y1-j,rad/2,rad/2);
fillellipse(x1+i,y1- j,rad/2,rad/2);
delay(10); cleardevice(); }
getch();
}
```

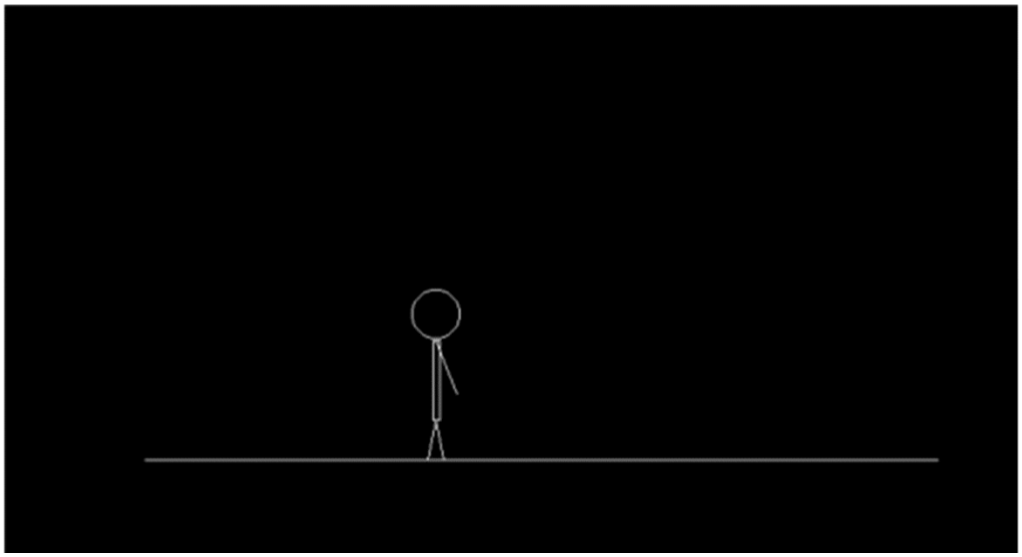






13. **Program to make a movingman**

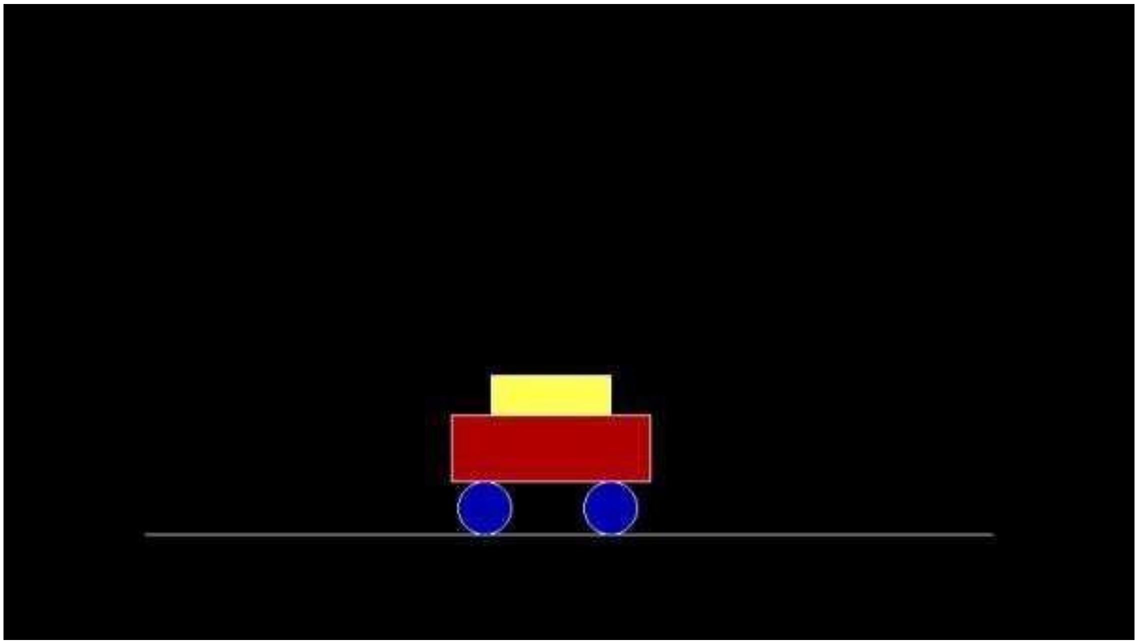
```
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
   int gd=DETECT,gm;
  initgraph(&gd,&gm,"C:\\TURBOC3\\bgi");
  for (int i=0;i<520;i=i+10)
  {
for(int  j=0;j<25;j++)
 {
  circle(50+i,300,18);
  line(0,410,600,410);
  rectangle(48+i,320,53+i,380)
  ; line(50+i,320,70+i-j,360);
  line(50+i,380,40+i+j,410);
  line(50+i,380,60+i-j,410);
  delay(10);
  clearviewport();
  } delay(10);
  clearviewport();
 }
  getch();
}
```



14.Program to create moving car scene

```
  #include<conio.h>
  #include<graphics.h>
  #include<dos.h>
 void main()
  {
  int gd=DETECT,gm,i;
  initgraph(&gd,&gm,"c:\\turboc3\\bgi");
  for(i=0;i<600;i++)
  {
  setfillstyle(SOLID_FILL,RED);
  rectangle(0+i,310,150+i,360);
  floodfill(10+i,320,WHITE);
  setfillstyle(SOLID_FILL,YELLOW
  ); rectangle(30+i,280,120+i,310);
  floodfill(40+i,285,WHITE);
  setfillstyle(SOLID_FILL,BLUE);
  circle(25+i,380,20);
  floodfill(25+i,380,WHITE);
  setfillstyle(SOLID_FILL,BLUE);
  circle(120+i,380,20);
  floodfill(120+i,380,WHITE);
  line(0,400,800,400);
  delay(10);
  clearviewport();
  }
  getch();
  }
```

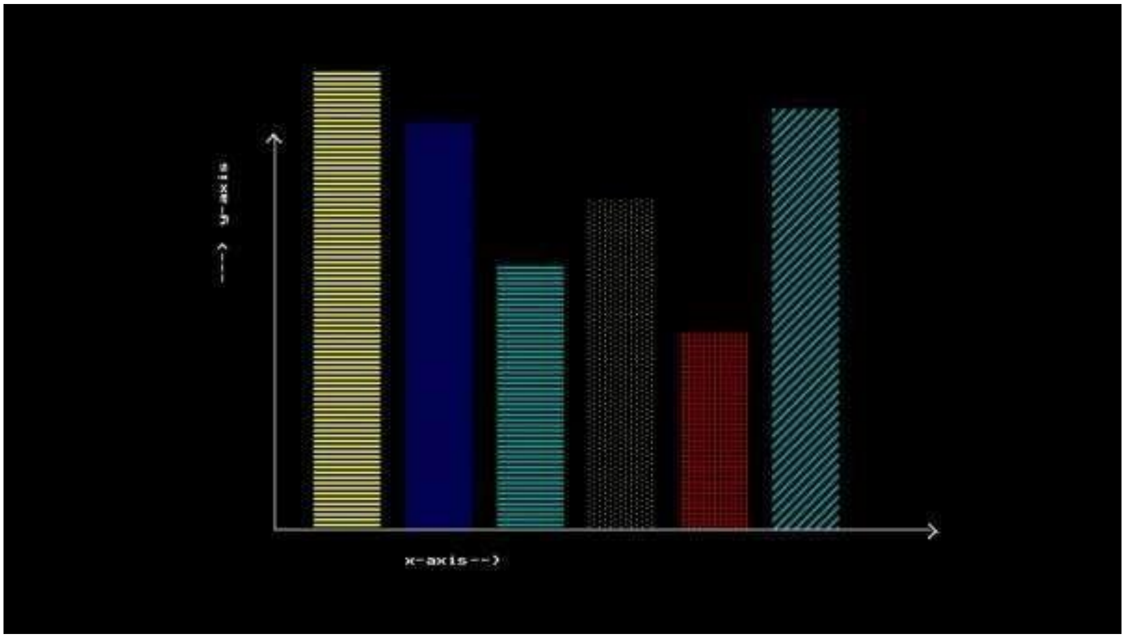
OUTPUT :



15. Program to draw bar graph

```
#include<conio.h>
#include<graphics.h>
 void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
line(100,100,100,400);//x axis
outtextxy(94,99,"/");
  outtextxy(98,99,"\\");
  line(100,400,600,400);//y axis
  outtextxy(599,396,"\\");
  outtextxy(599,401,"/");

  settextstyle(0,1,1); outtextxy(65,120,"y-
  axis");
outtextxy(65,180,"--->");
settextstyle(0,0,1);
outtextxy(200,420,"x- axis");
outtextxy(250,420,"-->");
setfillstyle(LINE_FILL,YELLOW);
bar(130,50,180,399);
  setfillstyle(INTERLEAVE_FILL,BLUE);
  bar(200,90,250,399);
  setfillstyle(LINE_FILL,CYAN);
  bar(270,200,320,399);
  setfillstyle(WIDE_DOT_FILL,YELLOW);
  bar(340,150,390,399);
  setfillstyle((HATCH_FILL,RED);
  bar(410,250,460,399);
  setfillstyle(SLASH_FILL,CYAN);
  bar(480,80,530,399);
 getch();

  }
```

OUTPUT :


---> y-axis
x-axis-->

16. Program to draw snooker table

```
  #include<conio.h>
  #include<graphics.
  h>
  #include<dos.h>
  void main() {
  Clrscr();
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\bgi");
for(int i=0;i<20;i++) {
setfillstyle(SOLID_FILL,GREEN);
rectangle(200,100,500,300);
rectangle(190,90,510,310);
floodfill(210,110,15);
setfillstyle(SOLID_FILL,RED);
floodfill(195,95,15);
setfillstyle(SOLID_FILL,BLACK);
circle(215,115,10); floodfill(220,120,15);
circle(355,115,10); floodfill(360,120,15);
circle(485,115,10); floodfill(490,120,15);
setfillstyle(SOLID_FILL,BLACK);
circle(215,285,10); floodfill(220,280,15);
circle(355,285,10); floodfill(360,280,15);
circle(485,285,10); floodfill(490,280,15);
setfillstyle(SOLID_FILL,BROWN);
```

```
rectangle(450+i,200,550+i,210);
```

```
floodfill(455+i,205,15);
floodfill(545+i,205,15);
floodfill(525+i,205,15);
setfillstyle(SOLID_FILL,WHITE);
circle(440,200,5);
floodfill(441,203,15);delay(100);
clearviewport(); }
for(int j=0;j<25;j++) {
setfillstyle(SOLID_FILL,GREEN)
; rectangle(200,100,500,300);
rectangle(190,90,510,310);
floodfill(210,110,15);
setfillstyle(SOLID_FILL,RED);
floodfill(195,95,15);
setfillstyle(SOLID_FILL,BLAC
K)
; circle(215,115,10);
floodfill(220,120,15);
circle(355,115,10);
floodfill(360,120,15);
circle(485,115,10);
floodfill(490,120,15);
setfillstyle(SOLID_FILL,BLACK)
; circle(215,285,10);
floodfill(220,280,15)
```

```
;circle(355,285,10);

floodfill(360,280,15)

;
```

```
circle(485,285,10);

floodfill(490,280,15);

setfillstyle(SOLID_FILL,BROWN

); rectangle(470-j,200,570-

j,210); floodfill(475-j,205,15);

floodfill(565-j,205,15);

floodfill(545-j,205,15);

setfillstyle(SOLID_FILL,WHITE)

; circle(440,200,5);

floodfill(441,203,15);

delay(100); clearviewport();

   }
for(int k=0;k<87;k++) {

setfillstyle(SOLID_FILL,GREEN

);rectangle(200,100,500,300);

rectangle(190,90,510,310);

floodfill(210,110,15);

setfillstyle(SOLID_FILL,RED);

floodfill(195,95,15);

setfillstyle(SOLID_FILL,BLACK);

circle(215,115,10);

floodfill(220,120,15);

circle(355,115,10);

floodfill(360,120,15);

circle(485,115,10);

floodfill(490,120,15);
```

```
  setfillstyle(SOLID_FILL,BLACK
  );circle(215,285,10);
  floodfill(220,280,15);
  circle(355,285,10);
  floodfill(360,280,15);
  circle(485,285,10);
  floodfill(490,280,15);
  setfillstyle(SOLID_FILL,BROWN)
  ; rectangle(445,200,545,210);
  floodfill(450,205,15);
  floodfill(540,205,15);
  floodfill(520,205,15);
  setfillstyle(SOLID_FILL,WHITE
  ); circle(440-k,200-k,5);
  floodfill(441-k,203-k,15);
  delay(50); clearviewport();
    }
}
```

OUTPUT :