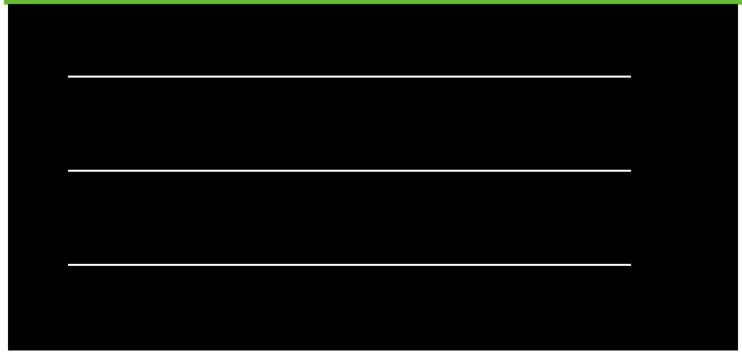
1)C Program to Draw a Pixel in Graphics

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main( )
{
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "C:\\TC\\BGI");
  putpixel(100,100,RED); //Putpixel(X1,Y1,Color)
  getch();
  closegraph();
}
```

3)

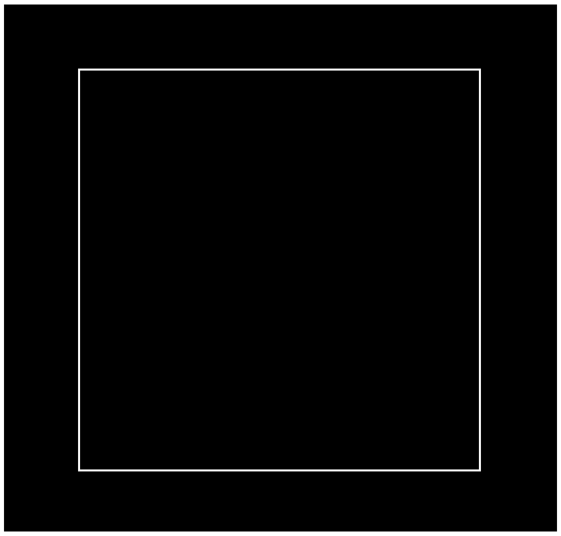
```
// C++ Implementation for drawing line
#include <graphics.h>
int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    line(150, 150, 450, 150);
    line(150, 200, 450, 200);
    line(150, 250, 450, 250);
    getch();
    closegraph();
}
```



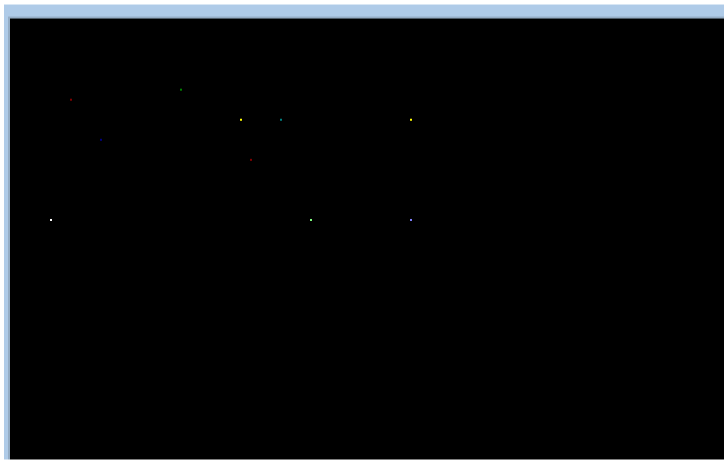
OUTPUT

7) WAP to draw a Rectangle by using rectangle function.

```
#include <graphics.h>

int main()
{
    int gd = DETECT, gm;
    int left = 150, top = 150;
    int right = 450, bottom = 450;
    initgraph(&gd, &gm, "");
    rectangle(left, top, right, bottom);
    getch();
    closegraph();

    return 0;
}
```

output 

```
    8)// C Implementation for putpixel()
#include <graphics.h>
#include <stdio.h>
int main()
{
    int gd = DETECT, gm, color;
    initgraph(&gd, &gm, "");
    putpixel(85, 35, GREEN);
    putpixel(30, 40, RED);
    putpixel(115, 50, YELLOW);
    putpixel(135, 50, CYAN);
    putpixel(45, 60, BLUE);
    putpixel(20, 100, WHITE);
    putpixel(200, 100, LIGHTBLUE);
    putpixel(150, 100, LIGHTGREEN);
    putpixel(200, 50, YELLOW);
    putpixel(120, 70, RED);

    getch();
    closegraph();

    return 0;
}
```

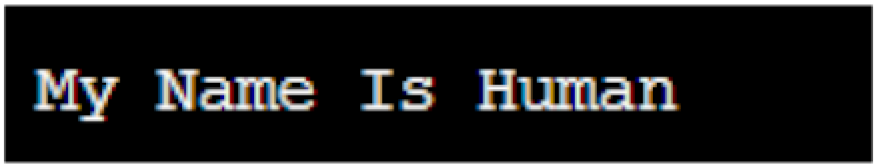


output

9) WAP to write your name at any specific position on the screen.

```
 #include<stdio.h>
void main()
{
     printf( " My Name Is Human " );
     getch();
}
```

Output :



```
My Name Is Human
```

10)

WAP to illustrate the working of setbkcolor function.

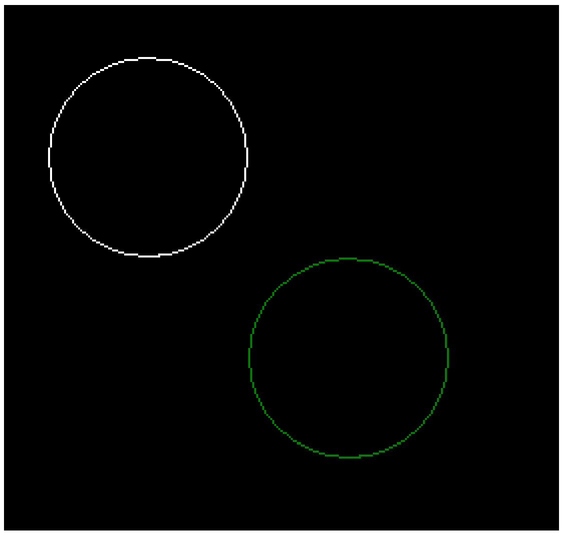
```
#include<graphics.h>
#include<conio.h>
main()
{
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\TC\\BGI");

   outtext("Press any key to change the background color to
GREEN.");
   getch();

   setbkcolor(GREEN);

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

11)

WAP to illustrate the functionality of setcolor function.

```
#include <graphics.h>
#include <stdio.h>
int main()
```

```
{
    int gd = DETECT, gm, color;
    initgraph(&gd, &gm, "");
    circle(100, 100, 50);
    setcolor(GREEN);
    circle(200, 200, 50);

    getch();


    return 0;
}
```

Output 

12)

WAP to illustrate the working of delay function.

```
#include <stdio.h>
// To use time library of C
#include <time.h>

void delay(int number_of_seconds)
{
    // Converting time into milli_seconds
    int milli_seconds = 1000 * number_of_seconds;
    // Storing start time
    clock_t start_time = clock();

    // looping till required time is not achieved
    while (clock() < start_time + milli_seconds)
        ;
}

// Driver code to test above function
int main()
{
    int i;
    for (i = 0; i < 10; i++) {
        // delay of one second
        delay(1);
        printf("%d seconds have passed\n", i + 1);
    }
    return 0;
}
```

13) WAP to illustrate the working of sleep function.

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

int main()
{

    printf("Program to sleep for 10 second in Windows.\n");

    Sleep(10);

    printf("This line will be executed after 10 millisecond.");

    return 0;
}
```

Output

```
Program to sleep for 10 second in Windows

This line will be executed after 10 millisecond
```

16)WAP to draw a circle.

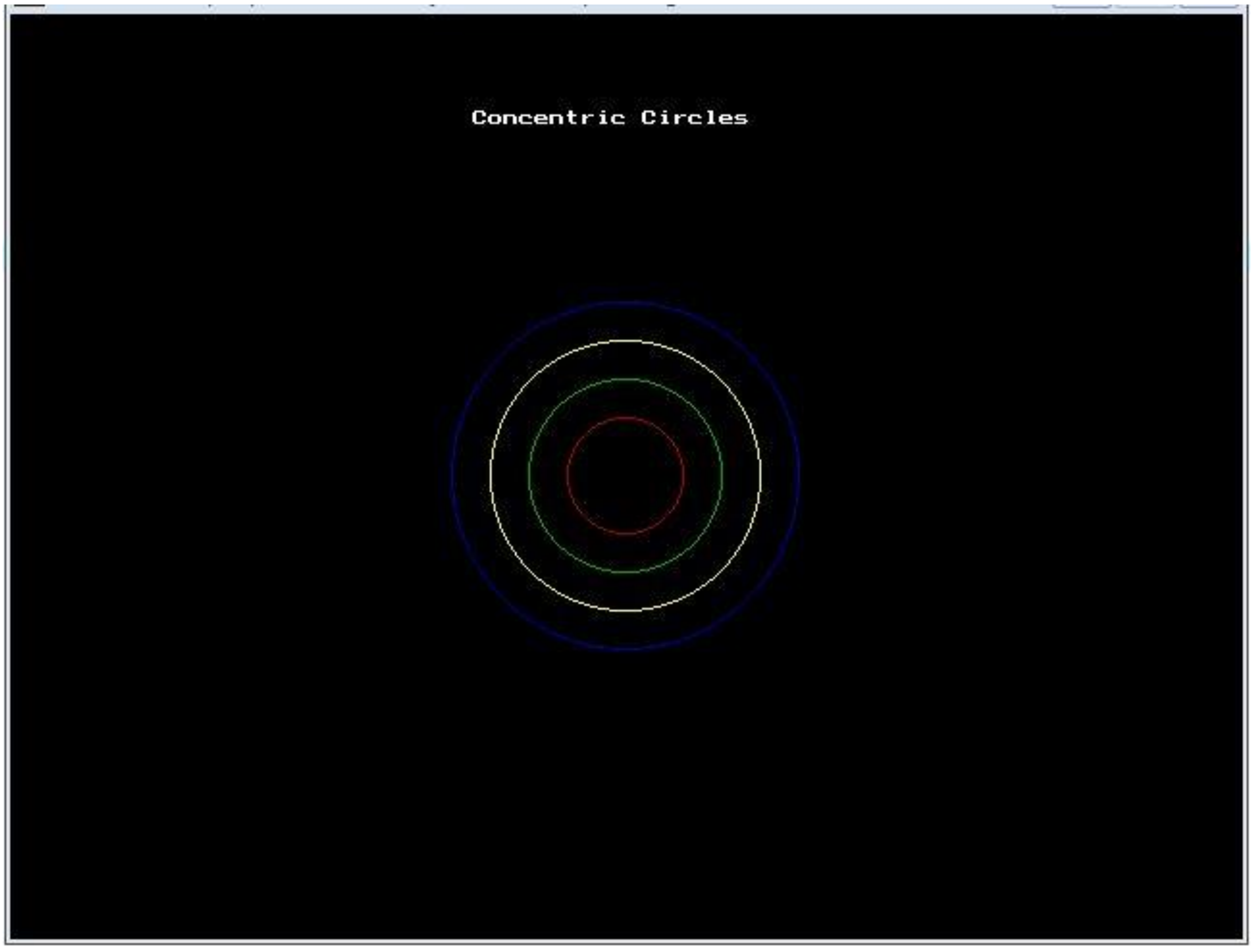
```c
// C Implementation for drawing circle
#include <graphics.h>

//driver code
int main()
{

    int gd = DETECT, gm;


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

    // circle function
    circle(250, 200, 50);

    getch();
    closegraph();

    return 0;
}
```

17)

WAP to draw 3 inner circles.

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

int main(){
   int gd = DETECT,gm;
   int x ,y;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   /* Initialize center of circle with center of screen */
   x = getmaxx()/2;
   y = getmaxy()/2;
```

```
    outtextxy(240, 50, "Concentric Circles");
    /* Draw circles on screen */
    setcolor(RED);
    circle(x, y, 30);
    setcolor(GREEN);
    circle(x, y, 50);
    setcolor(YELLOW);
    circle(x, y, 70);
    setcolor(BLUE);
    circle(x, y, 90);

    closegraph();
    return 0;
}
```



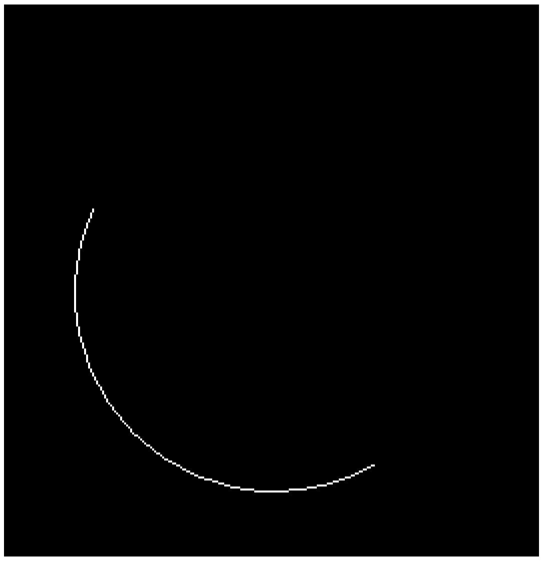
```
Concentric Circles
```

18)

WAP to draw an arc.

```
#include <graphics.h>

// driver code
int main()
{

    int gd = DETECT, gm;

    // location of the arc
    int x = 250;
    int y = 250;


    int start_angle = 155;
    int end_angle = 300;

    // radius of the arc
    int radius = 100;

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

    // arc function
    arc(x, y, start_angle, end_angle, radius);

    getch();

    closegraph();

    return 0;
}
```

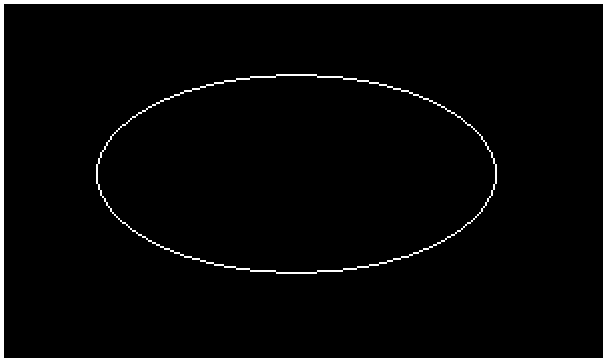
OUTPUT



19.	WAP to draw an ellipse.

```
#include <graphics.h>

int main()
{



    int x = 250, y = 200;


    int start_angle = 0;
    int end_angle = 360;

    // radius from x axis and y axis
    int x_rad = 100;
    int y_rad = 50;

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

    // ellipse function
    ellipse(x, y, start_angle,
     end_angle, x_rad, y_rad);
    getch();


    closegraph();

    return 0;
}
```

OUTPUT



20.	WAP to show the functionality of setfillstyle function.

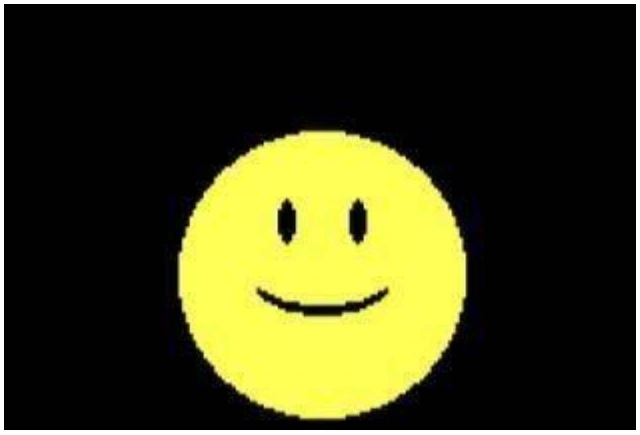
```
#include<graphics.h>
#include<conio.h>
main()
{
   int gd = DETECT, gm;

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

   setfillstyle(XHATCH_FILL, RED);
   circle(100, 100, 50);
   floodfill(100, 100, WHITE);

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

21 WAP to draw a smily face.

```
// C program to create a smiley face
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <stdio.h>

// Driver Code
int main()
{

        // Initialize graphic driver
        int gr = DETECT, gm;

        // Initialize graphics mode by passing
        // three arguments to initgraph function

        // &gdriver is the address of gdriver
        // variable, &gmode is the address of
        // gmode and "C:\\Turboc3\\BGI" is the
        // directory path where BGI files
        // are stored
        initgraph(&gr, &gm, "C:\\Turboc3\\BGI");

        // Set color of smiley to yellow
        setcolor(YELLOW);
```

```
    // creating circle and fill it with
    // yellow color using floodfill.
    circle(300, 100, 40);
    setfillstyle(SOLID_FILL, YELLOW);
    floodfill(300, 100, YELLOW);

    // Set color of background to black
    setcolor(BLACK);
    setfillstyle(SOLID_FILL, BLACK);

    // Use fill ellipse for creating eyes
    fillellipse(310, 85, 2, 6);
    fillellipse(290, 85, 2, 6);

    // Use ellipse for creating mouth
    ellipse(300, 100, 205, 335, 20, 9);
    ellipse(300, 100, 205, 335, 20, 10);
    ellipse(300, 100, 205, 335, 20, 11);

    getch();

    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system
    closegraph();

    return 0;
}
```

Output



22.	WAP to show the functionality of clear device function.

```
#include <graphics.h>

// driver code
int main()
{
    // gm is Graphics mode which is
    // a computer display mode that
    // generates image using pixels.
    // DETECT is a macro defined in
    // "graphics.h" header file
    int gd = DETECT, gm;

    // initgraph initializes the
    // graphics system by loading a
    // graphics driver from disk
    initgraph(&gd, &gm, "");

    // set the background colour as GREEN
    setbkcolor(GREEN);

    // outtext function displays
    // text at current position.
    outtext("Press any key to clear the screen.");
    getch();

    // cleardevice function
    cleardevice();

    outtext("Press any key to exit...");

    getch();

    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system .
    closegraph();

    return 0;
}
#include <graphics.h>

// driver code
int main()
{
    // gm is Graphics mode which is
    // a computer display mode that
    // generates image using pixels.
    // DETECT is a macro defined in
    // "graphics.h" header file
    int gd = DETECT, gm;

    // initgraph initializes the
    // graphics system by loading a
    // graphics driver from disk
    initgraph(&gd, &gm, "");

    // set the background colour as GREEN
    setbkcolor(GREEN);

    // outtext function displays
    // text at current position.
    outtext("Press any key to clear the screen.");

    getch();

    // cleardevice function
    cleardevice();

    outtext("Press any key to exit...");

    getch();

    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system .
    closegraph();

    return 0;
}
```

23)WAP to implement DDA algorithm.

```
#include <graphics.h>
#include <math.h>
#include <stdio.h>
int abs(int n) { return ((n > 0) ? n : (n * (-1))); }

void DDA(int X0, int Y0, int X1, int Y1)
{
        int dx = X1 - X0;
        int dy = Y1 - Y0;
        int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
        float Xinc = dx / (float)steps;
        float Yinc = dy / (float)steps;
        float X = X0;
        float Y = Y0;
        for (int i = 0; i <= steps; i++) {
                putpixel(round(X), round(Y),
                                 RED); // put pixel at (X,Y)
                X += Xinc;
                Y += Yinc;
                delay(100);

        }
}

// Driver program
int main()
{
        int gd = DETECT, gm;

        // Initialize graphics function
        initgraph(&gd, &gm, "");

        int X0 = 2, Y0 = 2, X1 = 14, Y1 = 16;

        // Function call
        DDA(2, 2, 14, 16);
        return 0;
}
```

Output

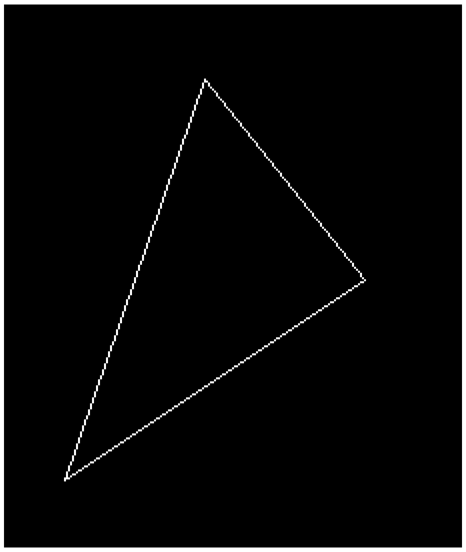
```
200 180
199 179
198 178
197 177
196 176
195 175
194 174
```

```
193 173

192 172

191 171

190 170

189 169

188 168

187 167

186 166

185 165

184 164

183 163

182 162

181 161
```

24)WAP to show draw poly function.

```
#include <graphics.h>

// driver code
int main()
{
    // gm is Graphics mode which is
    // a computer display mode that
    // generates image using pixels.
    // DETECT is a macro defined in
    // "graphics.h" header file
    int gd = DETECT, gm;

    // coordinates of polygon
    int arr[] = {320, 150, 400, 250,
                 250, 350, 320, 150};

    // initgraph initializes the
    // graphics system by loading a
    // graphics driver from disk
    initgraph(&gd, &gm, "");

    // drawpoly function
    drawpoly(4, arr);

    getch();

    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system .
    closegraph();

    return 0;
}
```

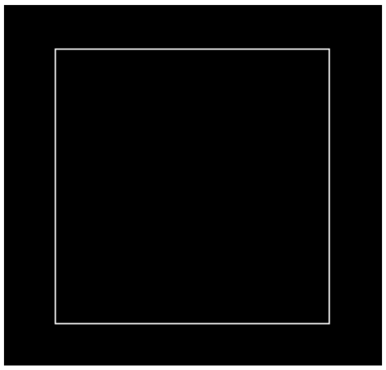
Output



25)
WAP to draw a Rectangle by using draw poly function.

```
#include <graphics.h>

// Driver code
int main()
{
    int gd = DETECT, gm;

    // location of left, top, right, bottom
    int left = 150, top = 150;
    int right = 450, bottom = 450;


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


    rectangle(left, top, right, bottom);

    getch();

    closegraph();

    return 0;
}
```

Output



26)WAP to draw a Triangle having all the 3 sides of 3 different colors.

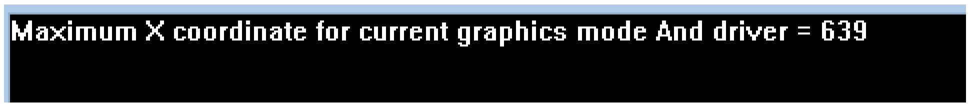
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>

void main() {
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "c:\\tc\\bgi");

  line(300, 100, 200, 200);
setfillcolor(3);
  line(300, 100, 400, 200);
setfillcolor(4);

  line(200, 200, 400, 200);
setfillcolor(5);

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

28)WAP to to show the functionality of getmaxx or getmaxy function.

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

// driver code
int main()
{
    // gm is Graphics mode which is
    // a computer display mode that
    // generates image using pixels.
    // DETECT is a macro defined in
    // "graphics.h" header file
    int gd = DETECT, gm;
    char arr[100];

    // initgraph initializes the
    // graphics system by loading a
    // graphics driver from disk
```

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

    // sprintf stands for “String print”.
    // Instead of printing on console, it
    // store output on char buffer which
    // are specified in sprintf
    sprintf(arr, "Maximum X coordinate for current "
        "graphics mode And driver = %d", getmaxx());

    // outtext function displays text at
    // current position.
    outtext(arr);

    getch();

    // closegraph function closes the
    // graphics mode and deallocates
    // all memory allocated by
    // graphics system .
    closegraph();

    return 0;
}
```

Output

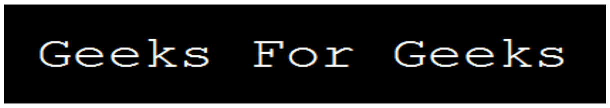


```
Maximum X coordinate for current graphics mode And driver = 639
```

29)WAP to show the functionality of settextstyle function.

```c
#include <graphics.h>

// driver code
int main()
{
    int gd = DETECT, gm;

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

    // location of text
    int x = 150;
    int y = 150;

    // font style
    int font = 8;

    // font direction
    int direction = 0;

    // font size
    int font_size = 5;

    // for setting text style
    settextstyle(font, direction, font_size);

    // for printing text in graphics window
    outtextxy(x, y, "Geeks For Geeks");

    getch();


    closegraph();

    return 0;
}
```

Output

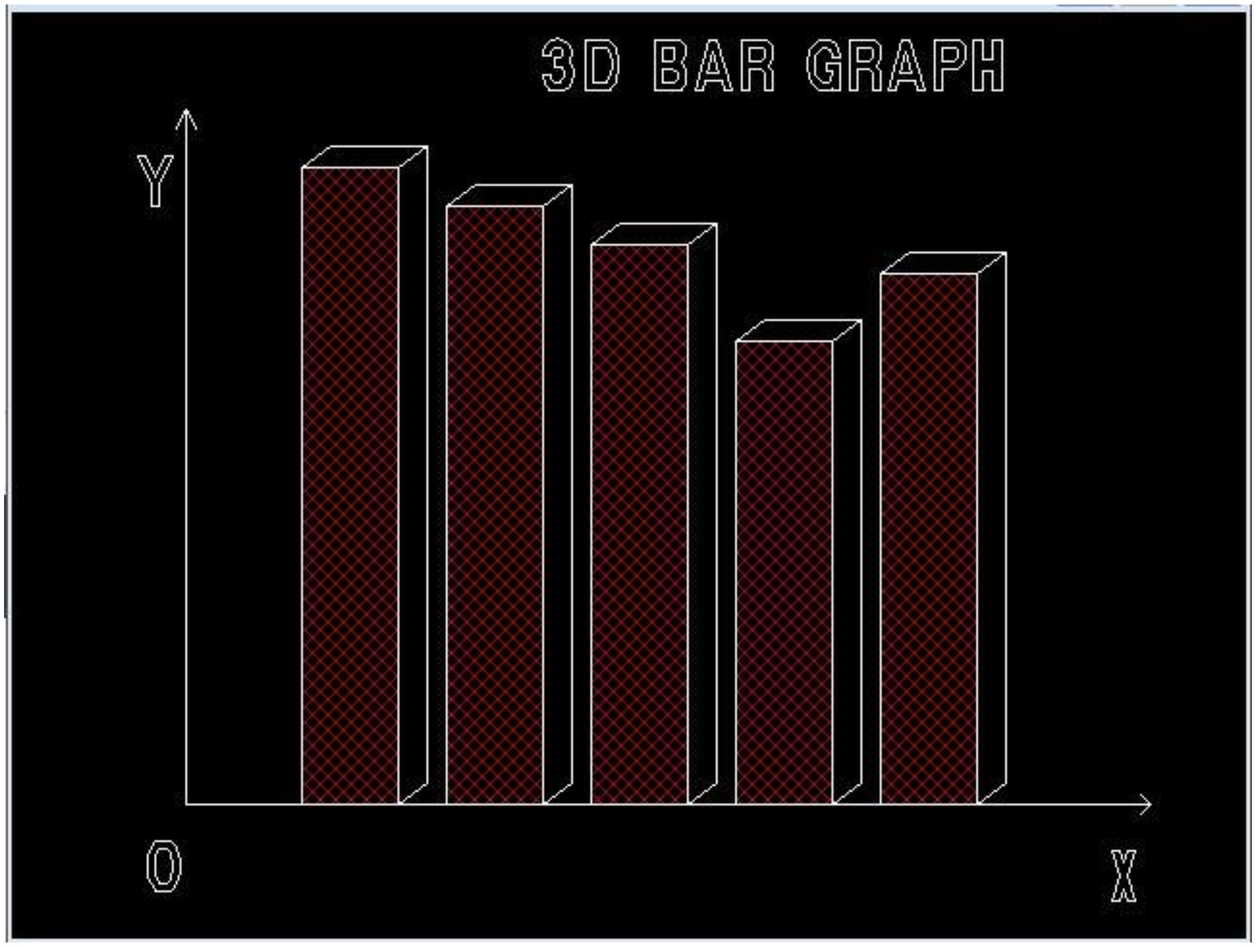


```
Geeks For Geeks
```

30) WAP to draw a 3D bar.

```
#include <graphics.h>

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "C:\\TC\\BGI");

   settextstyle(BOLD_FONT,HORIZ_DIR,2);
   outtextxy(275,0,"3D BAR GRAPH");

   setlinestyle(SOLID_LINE,0,2);
   /* Print X and Y Axis */
   line(90,410,90,50);
   line(90,410,590,410);
   line(85,60,90,50);
   line(95,60,90,50);
   line(585,405,590,410);
   line(585,415,590,410);

   outtextxy(65,60,"Y");
   outtextxy(570,420,"X");
   outtextxy(70,415,"O");

   /* Print 3D bars */
   setfillstyle(XHATCH_FILL, RED);
   bar3d(150,80,200,410, 15, 1);
   bar3d(225,100,275,410, 15, 1);
   bar3d(300,120,350,410, 15, 1);
   bar3d(375,170,425,410, 15, 1);
   bar3d(450,135,500,410, 15, 1);

   closegraph();
   return 0;
}
```

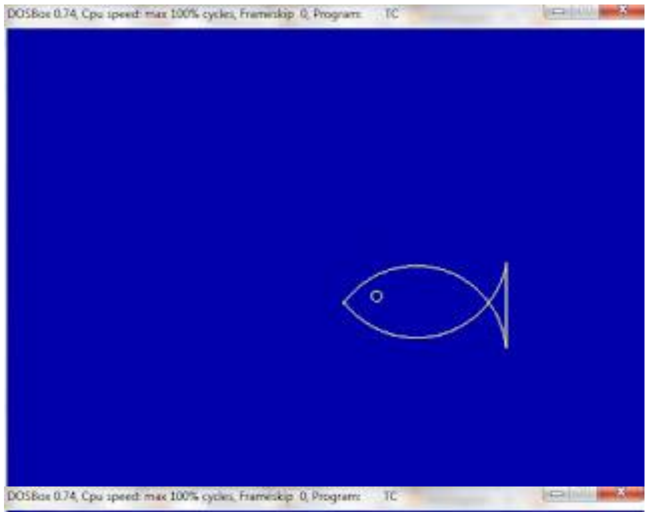
Output

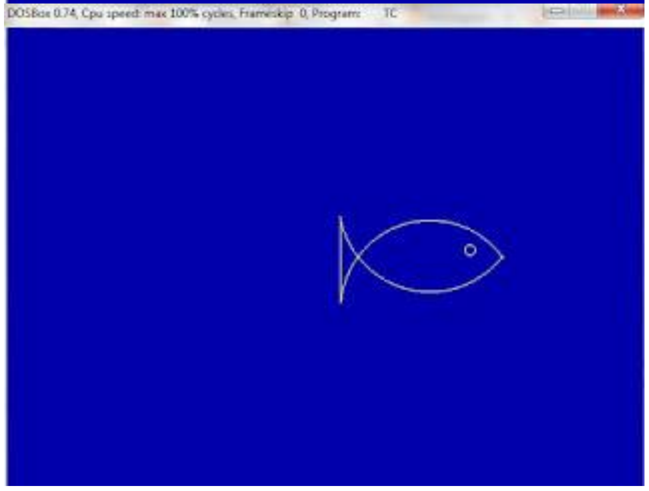


32.	WAP to draw an animated Fish.

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

void main()
{
 int gd=DETECT,gm,k,l,count=0;
 int x=10,y=200,x1=675,y1=380;
 int stangle=35,endangle=140,radius=90;

 initgraph(&gd,&gm,"c:\\TurboC3\\BGI");
  setbkcolor(BLUE);
 while(!kbhit())
 {
  cleardevice();

       setbkcolor(BLUE)
  if(x<640)
  {
   x+=5;
   y+=1;
   arc(x,y,stangle,endangle+35,radius);
   arc(x,y-110,190,323,radius+2);
   circle(x+40,y-60,5);
   line(x-90,y-90,x-90,y-8);
        }
  else
  {
   x1-=5;
   y1-=1;
   arc(x1,y1,stangle-30,endangle+4,radius);
   arc(x1,y1-110,217,350,radius+2);
   circle(x1-40,y1-60,5);
   line(x1+90,y1-90,x1+90,y1-10);
  }
  setcolor(YELLOW);
  delay(90);
 }
 closegraph();
}
```

Output

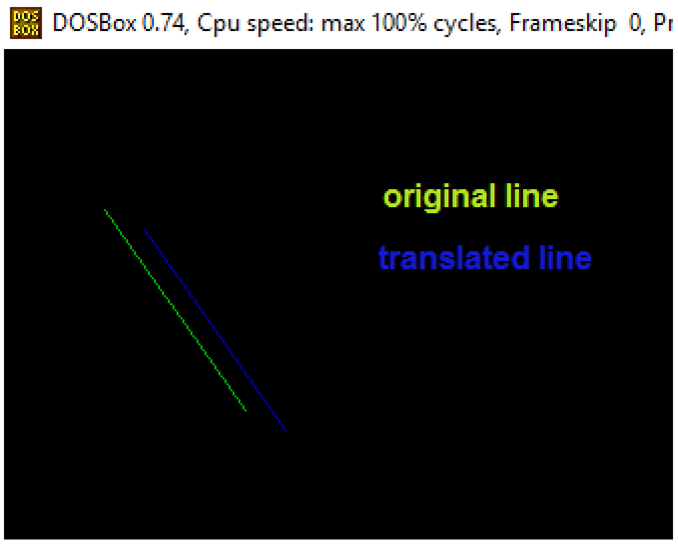

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC


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

33.	WAP to Translate an object.

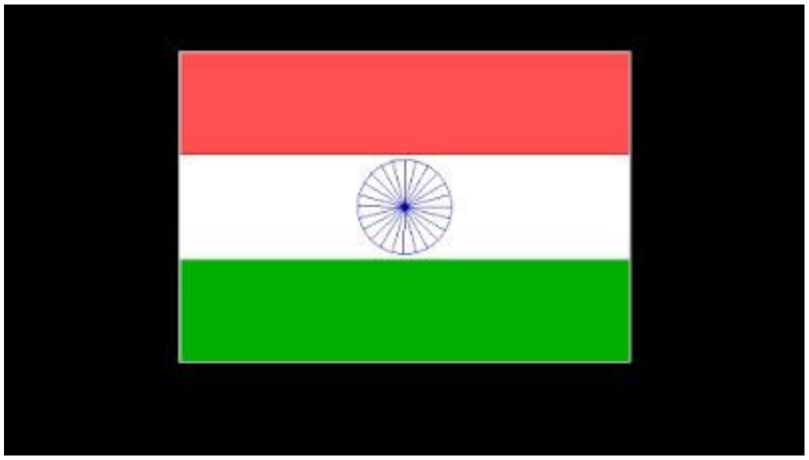
```
#include<bits/stdc++.h>
#include<graphics.h>

using namespace std;

// function to translate line
void translateLine ( int P[][2], int T[])
{
    /* init graph and line() are used for
       representing line through graphical
       functions
    */
    int gd = DETECT, gm, errorcode;
    initgraph (&gd, &gm, "c:\\tc\\bgi");

    // drawing original line using graphics functions
    setcolor (2);
    line(P[0][0], P[0][1], P[1][0], P[1][1]);

    // calculating translated coordinates
    P[0][0] = P[0][0] + T[0];
    P[0][1] = P[0][1] + T[1];
    P[1][0] = P[1][0] + T[0];
    P[1][1] = P[1][1] + T[1];

    // drawing translated line using graphics functions
    setcolor(3);
    line(P[0][0], P[0][1], P[1][0], P[1][1]);
    closegraph();
}

// driver program
int main()
{
    int P[2][2] = {5, 8, 12, 18}; // coordinates of point
    int T[] = {2, 1}; // translation factor
    translateLine (P, T);
    return 0;
}
```

Output



34)WAP to create Indian National Flag.

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

void main()
{
int i;
float x,y,PI=3.14;
int gd,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
rectangle(80,50,560,380);
line(80,160,560,160);
line(80,270,560,270);
setfillstyle(SOLID_FILL,LIGHTRED);
floodfill(81,51,WHITE);
setfillstyle(SOLID_FILL,WHITE);
floodfill(81,161,WHITE);
setfillstyle(SOLID_FILL,GREEN);
floodfill(81,271,WHITE);
setcolor(BLUE);
circle(320,215,50);

for(i=0;i<=360;i+=15)
{
x=50*cos(i*PI/180);
y=50*sin(i*PI/180);


line(320,215,320+x,215-y);
}
getch();
closegraph();
}
```



35.	WAP to draw a HUT.

```
#include<graphics.h>

int main(){
 int gd = DETECT,gm;
    initgraph(&gd, &gm, "X:\\TC\\BGI");
    /* Draw Hut */
    setcolor(WHITE);
    rectangle(150,180,250,300);
    rectangle(250,180,420,300);
    rectangle(180,250,220,300);

    line(200,100,150,180);
    line(200,100,250,180);
    line(200,100,370,100);
    line(370,100,420,180);

    /* Fill colours */
    setfillstyle(SOLID_FILL, BROWN);
    floodfill(152, 182, WHITE);
    floodfill(252, 182, WHITE);
    setfillstyle(SLASH_FILL, BLUE);
    floodfill(182, 252, WHITE);
    setfillstyle(HATCH_FILL, GREEN);
    floodfill(200, 105, WHITE);
    floodfill(210, 105, WHITE);

    closegraph();
    return 0;
}
```