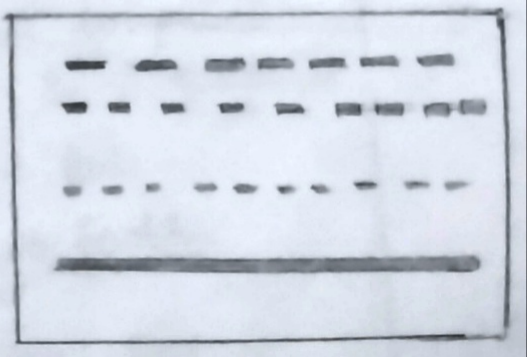
LAB PROGRAMS

① Write a program to display different pattern of lines.

①
```
# Include <graphics.h>
# Include <conio.h>
 Int main()
 {
 Int graphicdriver = DETECT, graphicmode;
 Initgraph (&graphicdriver, & graphicmode, "C:\\turboC3
                                          \\bgi");
outtextxy (20, 20+20, " Program to set different
 styles of lines in C graphics");
 int c, x = 120, y = 80;
 for ( c = 4; c >= 0; c--)
     {
      setlinestyle (c, 0, 3);
      line (x, y, x+150, y);
        y = y+20;
      }
     getch();
     return 0;
     }
```

Output :



② WAP to create various types of texts and fonts

②
```
# include < graphics.h >
# include < conio.h >
int main()
{
int graphicdriver = DETECT, graphicmode;
initgraph(&graphicdriver, &graphicmode, "c:\\turboc3
                                        \\bgi");
outtextxy(20, 20+20, "Program to print different
  fonts in C graphics");
int x = 75, y = 75, f = 0;
for(f = 0; f <= 5; f++)
  {
  settextstyle(f, HORIZ_DIR, 1);
  outtextxy(x, y, "Testing different fonts..");
   y = y + 20;
  }
  getch();
  return 0;
  }
```

Output :- Program to print different fonts in C graphics

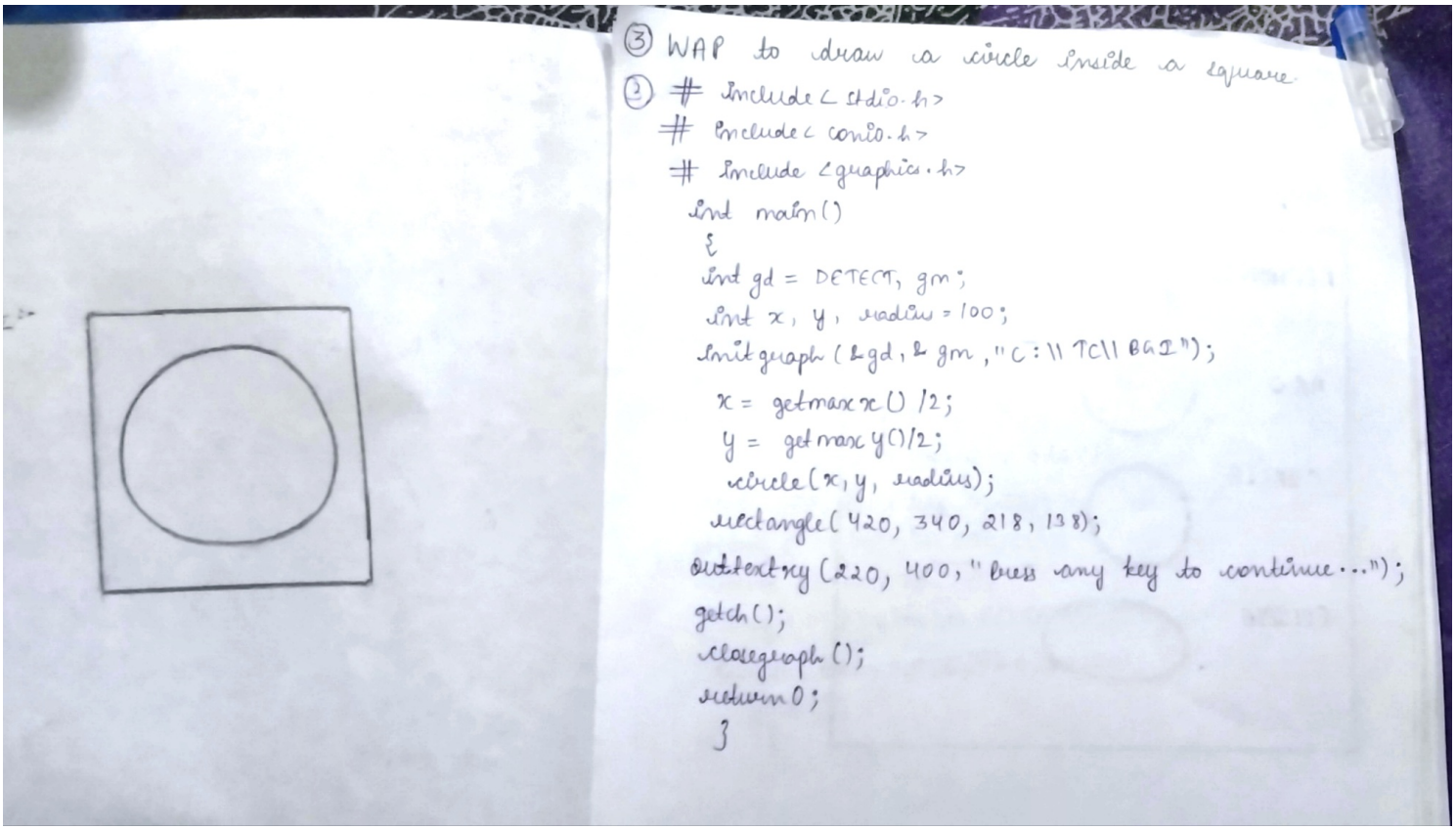
Testing different fonts..

Testing different fonts..

Testing different fonts

Testing different fonts..

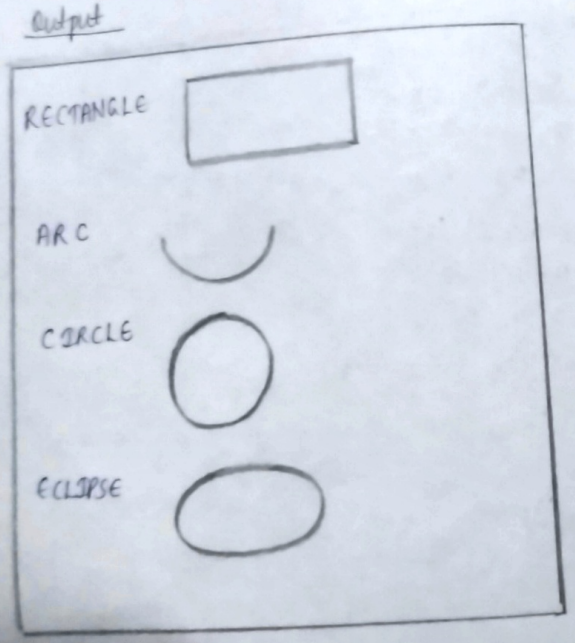
Testing different fonts.



③ WAP to draw a circle inside a square.

②
```
# Include < stdio.h >
# Include < conio.h >
# Include < graphics.h >
int main()
 {
 int gd = DETECT, gm;
 int x, y, radius = 100;
 initgraph ( &gd, &gm, "C:\\TC\\BGI");
  x = getmaxx() /2;
  y = getmaxy()/2;
  circle(x, y, radius);
 rectangle(420, 340, 218, 138);
outtextxy (220, 400, "Press any key to continue...");
getch();
closegraph();
return 0;
 }
```

Output



① WAP to draw different types of shapes(circle, arc, sector, ellipse, rectangle)

②
```
# include < graphics. h>
# include < conio. h>
 void main()
   {
 int gd = DETECT, gm;
 initgraph (& gd, & gm, "C:\\TC\\BGI");
 setbkcolor (GREEN);
 printf ("\t\t\n\n\n\n RECTANGLE");
 rectangle (125, 115, 215, 165);
 printf ("\t\t\n\n\n\n ARC");
  arc (120, 200, 180, 0, 30);
  printf ("\t\n\n\n CIRCLE");
  circle (120, 270, 30);
  printf ("\t\n\n\n ECLIPSE");
   ellipse (120, 350, 0, 360, 30, 20);
     getch();
     }
```

(5) WAP to implement DDA algorithm for drawing a line segment between two given end points (x_1, y_1) and (x_2, y_2).

(5)
```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
void main()
{
float x1, x, y, y1, y2, dx, dy, x2, step;
int i, gd = DETECT, gm;
initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
printf("Enter the value of x1 and y1:");
scanf("%f %f", &x1, &y1);
printf("Enter the value of x2 and y2:");
scanf("%f %f", &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+dx;
i = i+1;
delay(100);
}
closegraph();
}
```

Output :- Enter the value of x1 and y1 :

100 100

Enter the value of x2 and y2 :

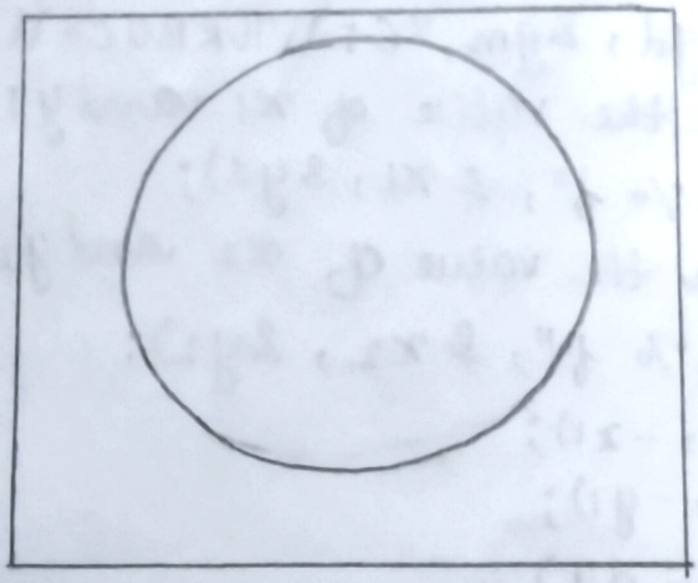
150 150

(6) WAP to Implement Bresenham's circle algorithm for drawing a circle of given center (x, y) and radius r.

(6)
```
# include <stdio.h>
# include <dos.h>
# include <graphics.h>
void drawcircle (int xc, int yc, int x, int y)
{
putpixel(xc+x, yc+y, RED);
putpixel(xc-x, yc+y, RED);
putpixel(xc+x, yc-y, RED);
putpixel(xc-x, yc-y, RED);
putpixel(xc+y, yc+x, RED);
putpixel(xc-y, yc+x, RED);
putpixel(xc+y, yc-x, RED);
putpixel(xc-y, yc-x, RED);
}
void circleBres(int xc, int yc, int r)
{
int x=0, y=r;
int d = 3-2*r;
drawcircle(xc, yc, x, y);
while (y>=x)
{
x++;
if (d>0)
{
y--;
d = d+4*(x-y)+10;
}
else
d = d+4*x+6;
drawcircle(xc, yc, x, y);
delay(50);
}}
```

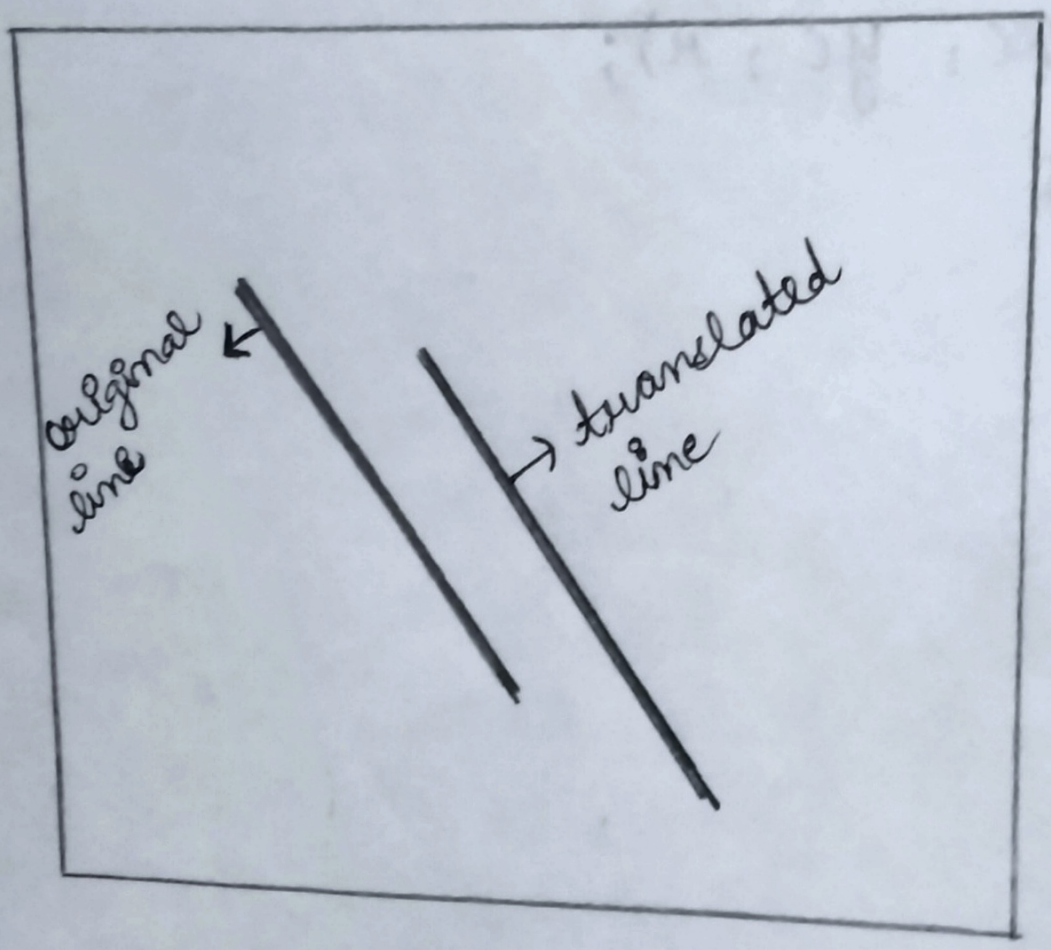
```
int main()
 {
int xc = 50, yc = 50, r = 30;
int gd = DETECT, gm;
initgraph(&gd, &gm, " ");
circleBres(xc, yc, r);
return 0;
 }
```

Output :-



⑧ WAP to perform Translation in 2D transformation

⑧
```
# include< iostream>
# include< graphics.h>
void translateline (int P[][2], int T[])
  {
int gd = DETECT, gm, errorcode;
initgraph (&gd, &gm, "C:\\TC\\BGI");
setcolor(2);
line [P[0][0], P[0][1], P[1][0], P[1][1]);
 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];
  setcolor(3);
  line (P[0][0], P[0][1], P[1][0], P[1][1]);
   closegraph();
    }
  int main()
   {
  int P[2][2] = {{5, 8}, {12, 18}};
  int T[] = {2, 1};
  translateline (P, T);
  return 0;
  }
```


original line
translated line

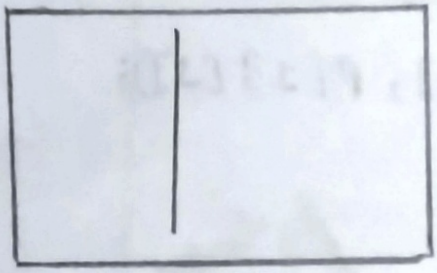
(10) WAP to perform Rotation in 2D transformation.

(10)
```
# include <stdio.h>
# include<graphics.h>
# include<math.h>
Int main()
{
int gd = 0, gm, x1, x2, y1, 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;
          }
```

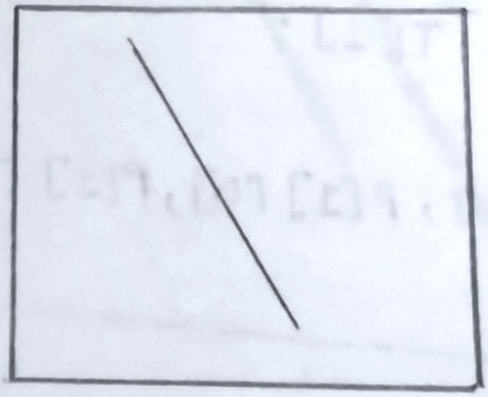
Output :- Enter coordinates of line:

100 100

100 200



Enter rotation angle : 45

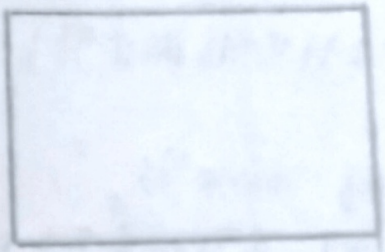


WAP for in graphics for X - shear.

```
#include<stdio.h>
#include<graphics.h>
void main()
{
int gd = DETECT, gm;
int x, y, x1, y1, x2, y2, x3, y3, shear_f;
initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
printf("Please enter first coordinate =");
scanf("%d %d", &x, &y);
printf("Please enter second coordinate =");
scanf("%d %d", &x1, &y1);
printf("Please enter third coordinate =");
scanf("%d %d", &x2, &y2);
printf("Please enter last coordinate =");
scanf("%d %d", &x3, &y3);
printf("Please enter shearing factor y =");
scanf("%d", &shear_f);
cleardevice();
line(x, y, x1, y1);
line(x1, y1, x2, y2);
line(x2, y2, x3, y3);
line(x3, y3, x, y);
setcolor(RED);
y = y + x * shear_f;
y1 = y1 + x1 * shear_f;
y2 = y2 + x2 * shear_f;
y3 = y3 + x3 * shear_f;
line(x, y, x1, y1);
line(x1, y1, x2, y2);
line(x2, y2, x3, y3);
closegraph();;
```

Output :-

Before shearing



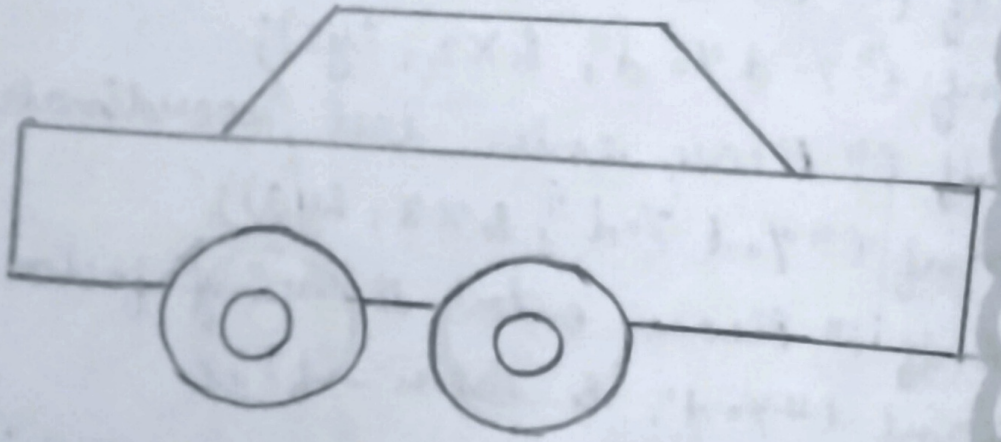
After shearing

WAP for creating a simple car shape

```
# include <graphics.h>
# include <stdio.h>
void draw_moving_car(void)
{
int i, j=0, gd = DETECT, gm;
initgraph (&gd, &gm, " ");
for(i=0; i<=420; i = i+10)
{
setcolor(RED);
line (0+i, 300, 210+i, 300);
line (50+i, 300, 75+i, 270);
line (75+i, 270, 150+i, 270);
line (150+i, 270, 165+i, 300);
line (0+i, 300, 0+i, 330);
line (210+i, 300, 210+i, 330);
circle(65+i, 330, 15);
circle(65+i, 330, 2);
circle (145+i, 330, 15);
circle(145+i, 330, 2);
line (0+i, 330, 50+i, 330);
line (80+i, 330, 130+i, 330);
line (210+i, 330, 160+i, 330);
delay (100);
setcolor (BLACK);
line(0+i, 300, 210+i, 300);
line (50+i, 300, 75+i, 270);
line (75+i, 270, 150+i, 270);
line (150+i, 270, 165+i, 300);
line (0+i, 300, 0+i, 330);
line (210+i, 300, 210+i, 330);
circle (65+i, 330, 15);
circle (65+i, 330, 2);
```

```
circle (145+i, 330, 15);
circle (145+i, 330, 2);
line (0+i, 330, 50+i, 330);
line (80+i, 330, 130+i, 330);
line (210+i, 330, 160+i, 330);
}
getch();
closegraph();
}
int main()
{
draw_moving_car();
return 0;
}
```

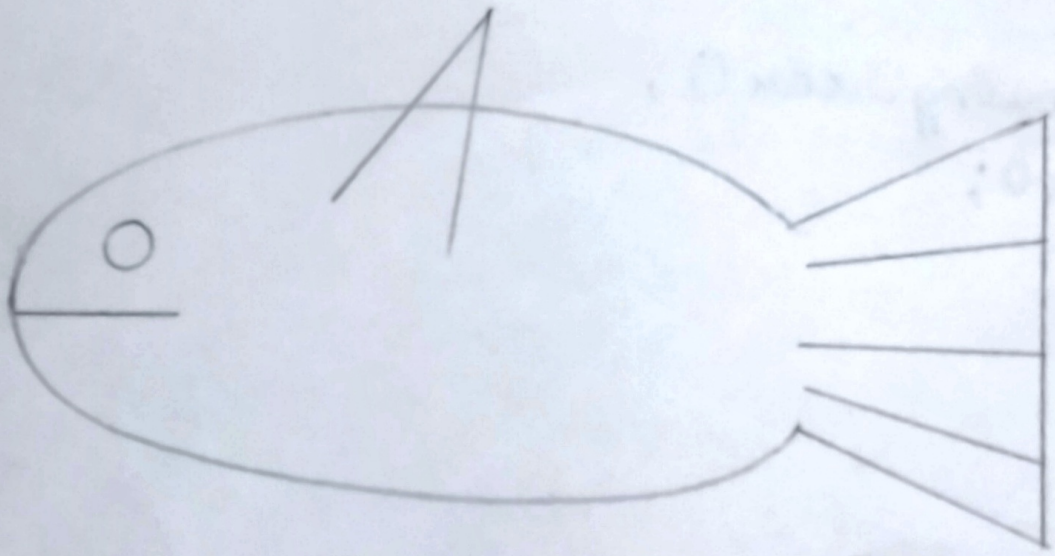
output :-



WAP in graphics for creating fish

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
void main()
{
int gd = DETECT, gm;
initgraph (&gd, &gm, "C:/TURBOCB/BGI");
ellipse (520, 200, 30, 330, 90, 30);
circle (450, 193, 3);
line (430, 200, 450, 200);
line (597, 185, 630, 170);
line (597, 215, 630, 227);
line (630, 170, 630, 227);
line (597, 200, 630, 200);
line (597, 192, 630, 187);
line (597, 207, 630, 213);
line (500, 190, 540, 150);
line (530, 190, 540, 150);
getch();
}
```

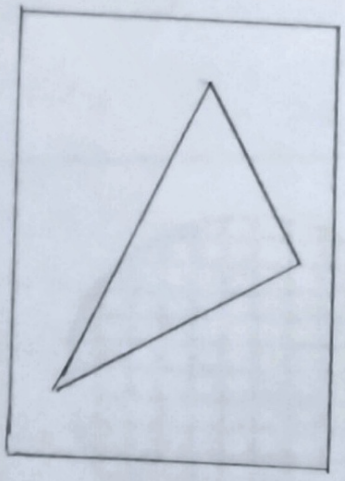
Output :-



(17) WAP that illustrating the use of fillpoly function.

(17)
```
#include<graphics.h>
#include<stdio.h>
int main()
{
int gd=DETECT, gm;
int arr[]={320,150,400,250,250,350,320,150};
initgraph(&gd,&gm," ");
fillpoly(5,arr);
getch();
closegraph();
return 0;
}
```

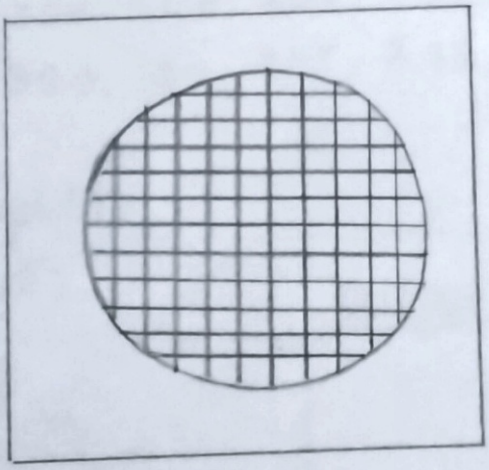
Output :-



(18) WAP that illustrating the use of floodfill function

(18)
```
#include<stdio.h>
#include<graphics.h>
int main()
{
int gd = DETECT, gm;
initgraph(&gd, &gm, " ");
int x_circle = 250;
int y_circle = 250;
int radius = 100;
int border_color = WHITE;
setfillstyle(HATCH_FILL, RED);
circle(x_circle, y_circle, radius);
floodfill(x_circle, y_circle, border_color);
getch();
closegraph();
return 0;
}
```

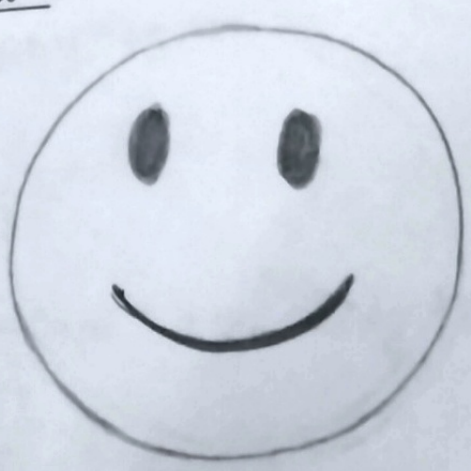
Output :-



(20) Write a smiling face program in C.

```
# include < graphics.h >
# include < conio.h >
# include < stdio.h >
# include < dos.h >
int main()
{
int gr = DETECT, gm;
initgraph (&gr, &gm, "C:\\TURBOC3\\BGI");
setcolor (YELLOW);
circle (300, 100, 40);
setfillstyle (SOLID_FILL, YELLOW);
floodfill (300, 100, YELLOW);
setcolor (BLACK);
setfillstyle (SOLID_FILL, BLACK);
fillellipse (310, 85, 2, 6);
fillellipse (290, 85, 2, 6);
ellipse (300, 100, 205, 335, 20, 9);
ellipse (300, 100, 205, 335, 20, 10);
ellipse (300, 100, 205, 335, 20, 11);
getch();
closegraph();
return 0;
}
```

Output :-



(21) Write a C program to create traffic simulation

(24)
```
#include <graphics.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
void main()
{
int gd = DETECT, gm, midx, midy;
initgraph (&gd, &gm, "C:\\TC\\BGI");
midx = getmaxx() /2;
midy = getmaxy () /2;
setcolor(CYAN);
settextstyle (TRIPLEX_FONT, HORIZ_DIR, 4);
settextjustify (CENTER_TEXT, CENTER_TEXT);
outtextxy (midx, midy - 10, "TRAFFIC LIGHT SIMULATION");
outtextxy (midx, midy + 10, "PRESS ANY KEY TO START");
getch();
cleardevice();
setcolor (WHITE);
settextstyle (DEFAULT_FONT, HORIZ_DIR, 1);
rectangle (midx - 30, midy - 80, midx + 30, midy + 80);
circle (midx, midy, - 50, 22);
setfillstyle (SOLID_FILL, RED);
floodfile (midx, midy - 50, 22);
setcolor (BLUE);
outtextxy (midx - 18, midy - 3, "READY");
delay (2000);
cleardevice();
setcolor (WHITE);
rectangle (midx - 30, midy - 80, midx + 30, midy + 80);
circle (midx, midy + 50, 22);
setfillstyle (SOLID_FILL, GREEN);
```

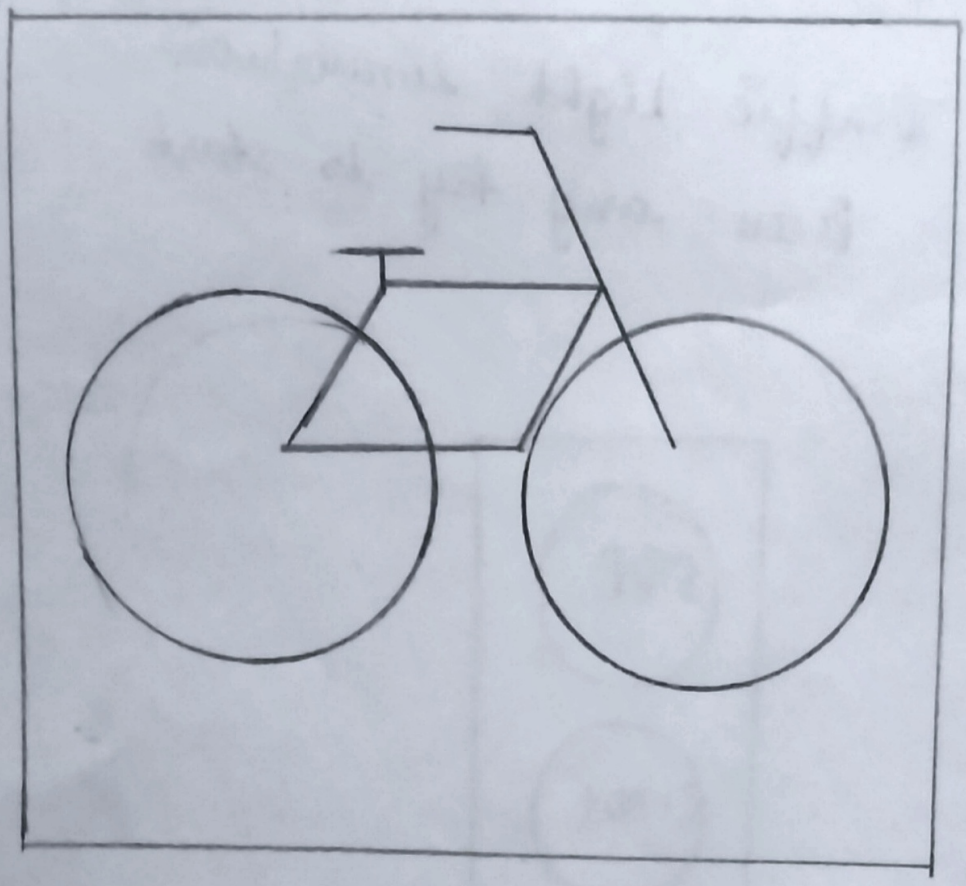
```
floodfill (midx, midy + 50, WHITE);
    setcolor (BLUE);
outtextxy( midx - 7, midy + 48, "GO");
    setcolor( RED);
settextstyle(TRIPLEX_FONT, HORIZ_DIR, 4);
outtextxy(midx - 150, midy + 100, "Brrrmmm");
    getch();
    }
```

Output :- Traffic Light simulation
Press any key to start

(24) WAP to draw the moving cycle using computer graphics.

(24)
```
# include < graphics . h >
# include < dos . h >
# include < conio . h >
# include < iostream . h >
int main()
{
int gd = DETECT, gm, i, a;
initgraph (& gd, & gm, "C:\\TURBO C3 \\BGI");
for (i = 0 ; i < 600 ; i++)
   {
line (50 + i, 405, 100 + i, 405);
line (75 + i, 375, 125 + i, 375);
line ( 50 + i, 405, 75 + i, 375);
line (100 + i, 405, 100 + i, 345);
line (150 + i, 405, 100 + i, 345);
line (75 + i, 345, 75 + i, 370);
line (70 + i, 370, 80 + i, 370);
line (80 + i, 345, 100 + i, 345);
circle (150 + i, 405, 30);
circle ( 50 + i, 405, 30);
line (0, 436, getmaxx(), 436);
rectangle ( getmaxx () - i, 436, 650 - i, 431);
  delay (10);
  cleardevice ();
   }
  getch ();
  closegraph ();
   }
```

Output :-



(26) WAP in C to make a digital clock.

(25)
```
# include < stdio.h >
# include < graphics.h >
# include < dos.h >
  void main()
  {
  for (int i=0; i<=24; i++)
   {
   for ( int j=0; j<=60; j++)
    {
    for(int k=0; k<=60; k++)
     {
     clrscr();
   printf (" [%d : %d : %d]", i, j, k);
    } } }
   getch();
   }
```

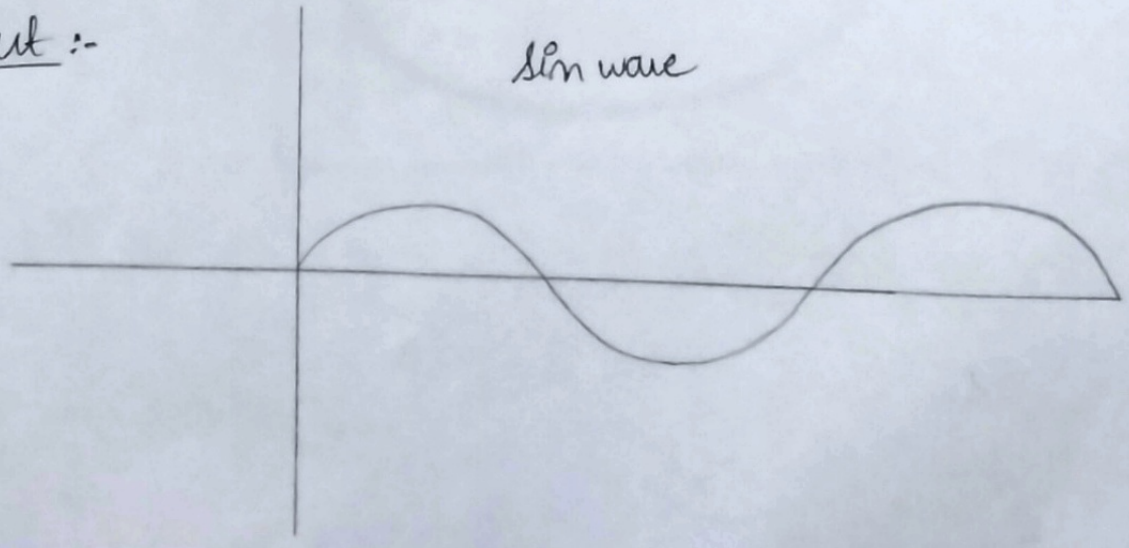
Output :-

11 : 45 : 03
hr min sec

(28) WAP in C to draw a sine wave using graphics.

(28)
```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
#include <math.h>
int main()
{
int gd = DETECT, gm;
int angle = 0;
double x, y;
initgraph(&gd, &gm, "C:\\TC\\BGI");
line(0, getmaxy()/2, getmaxx(), getmaxy()/2);
for(x=0; x<getmaxx(); x+=3);
{
y = 50 * sin(angle * 3.141/180);
y = getmaxy()/2 - y;
putpixel(x, y, 15);
delay(100);
angle += 5;
}
getch();
closegraph();
}
```

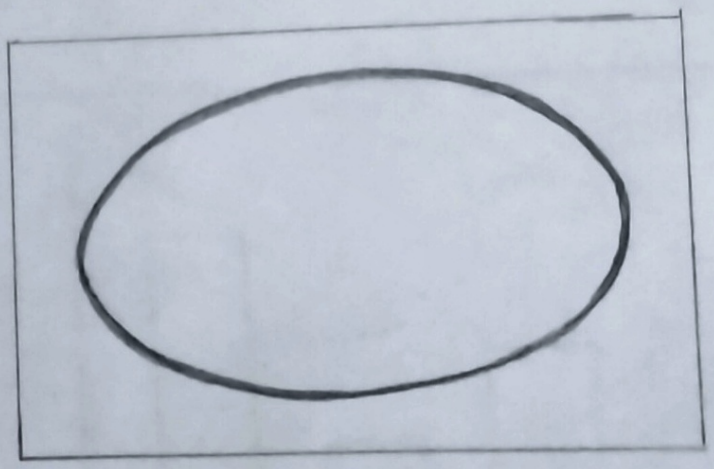
Output :-



(29) WAP in C to draw an eclipse using graphics.

(29)
```
#include<graphics.h>
#include<conio.h>
int main()
{
int gd = DETECT, gm;
int x = 250, y = 200;
int start_angle = 0;
int end_angle = 360;
int x_rad = 100;
int y_rad = 50;
initgraph(&gd, &gm, " ");
ellipse(x, y, start_angle, end_angle,
        x_rad, y_rad);
getch();
closegraph();
return 0;
}
```

Output :-

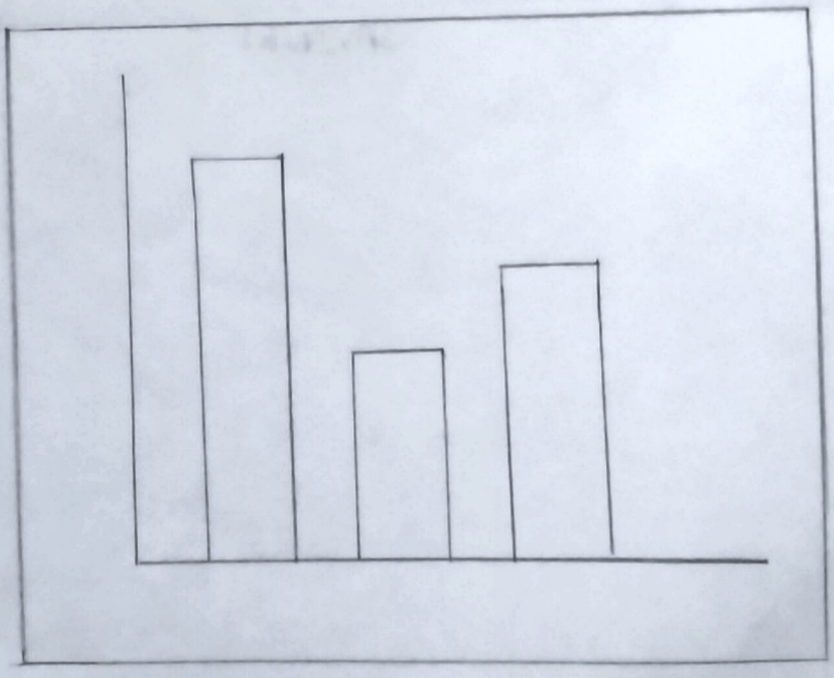


(30) WAP in C to draw bar graph using graphics.

(30)

```
#include<graphics.h>
#include<conio.h>
int main()
{
int gd = DETECT, gm;
initgraph(&gd, &gm, "");
int left, top, right, bottom;
bar(left = 150, top = 150, right = 190, bottom = 350);
bar(left = 220, top = 250, right = 260,
                           bottom = 350);
bar(left = 290, top = 200, right = 330,
                           bottom = 350);
line(100, 50, 100, 350);
line(100, 350, 400, 350);
getch();
closegraph();
}
```

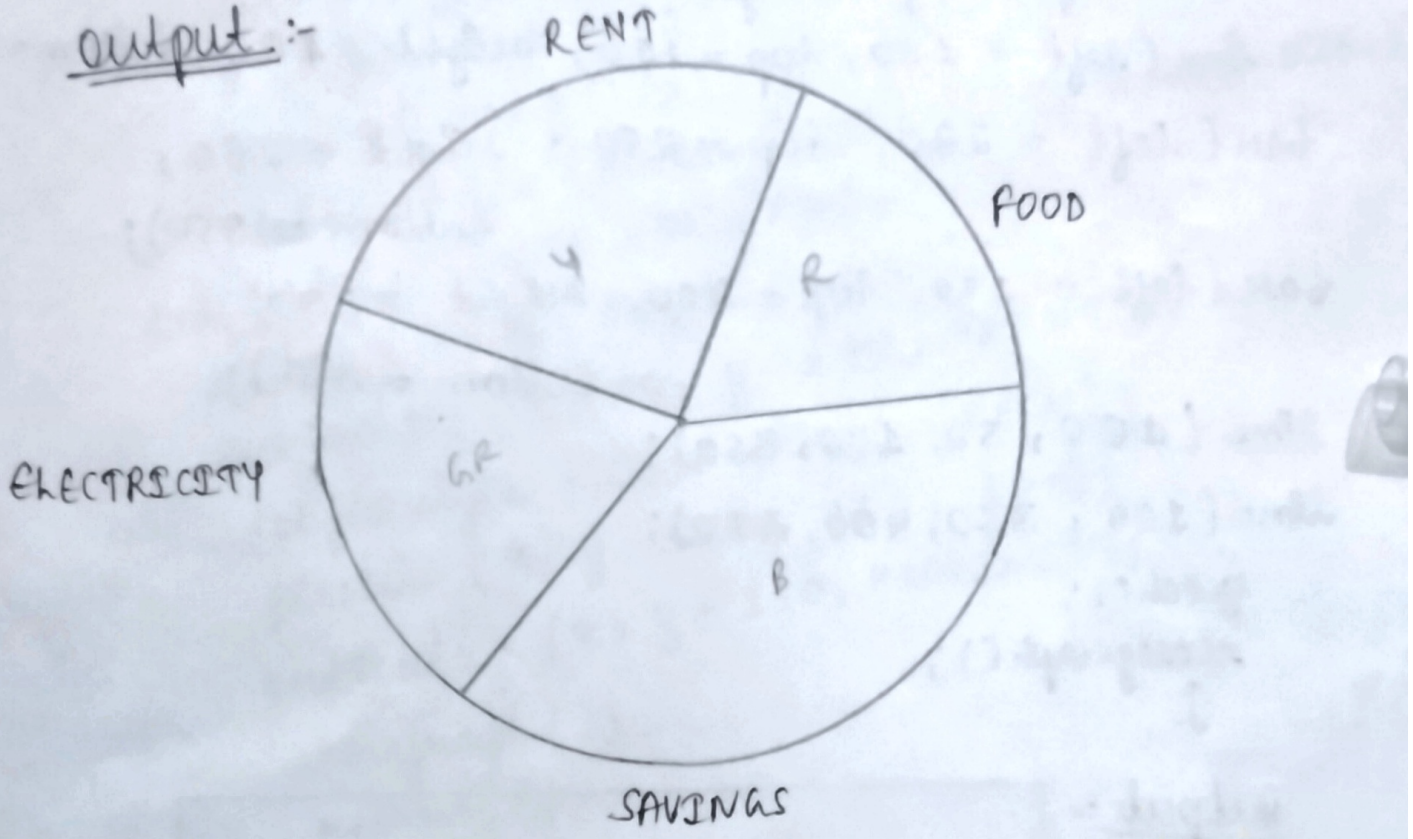
Output :-



(31) WAP in C to draw a pie chart using graphics

(A)
```
#include<graphics.h>
int main()
{
int gd = DETECT, gm, x, y;
initgraph(&gd, &gm, "C:\\TC\\BGI");
settextstyle(BOLD_FONT, HORIZ_DIR, 2);
outtextxy(220, 10, "PIE CHART");
x = getmaxx()/2;
y = getmaxy()/2;
settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 1);
setfillstyle(SOLID_FILL, RED);
pieslice(x, y, 60, 160, 120);
outtextxy(x-30, y-170, "RENT");
setfillstyle(SOLID_FILL, GREEN);
pieslice(x, y, 160, 220, 120);
outtextxy(x, y+150, "SAVINGS");
closegraph();
return 0;
}
```

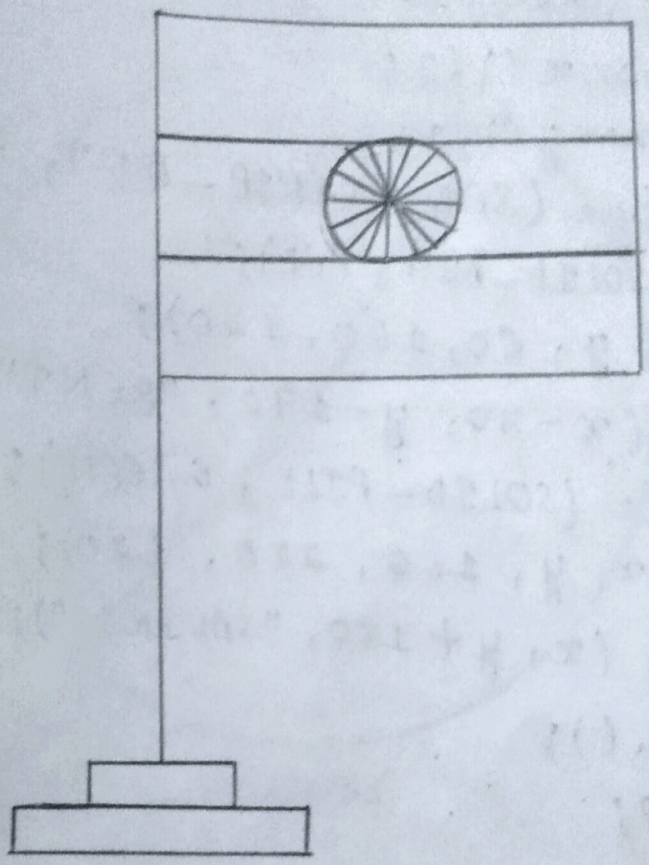
Output :-



(32) WAP in C to draw national flag using graphics

(32)
```
# include < conio.h >
# include < graphics.h >
# include < stdio.h >
  void main()
   {
  initgraph (&gd, &gm, "C:\\TURBOC3\\BGI");
  line (250, 100, 250, 600);
  line (250, 100, 250, 600);
  setfillstyle (SOLID_FILL, WHITE);
  rectangle (225, 600, 275, 610);
  rectangle (200, 610, 300, 620);
  floodfill (227, 608, 15);
   floodfill (202, 618, 15);
  setfillstyle (SOLID_FILL, LIGHTRED);
  rectangle (250, 100, 650, 280);
  line (250, 160, 650, 160);
  floodfill (252, 158, 15);
   setfillstyle (SOLID_FILL, BLUE);
    circle (450, 190, 30);
    floodfill (452, 188, 15);
    setfillstyle (SOLID_FILL, WHITE);
    line (250, 160, 480, 160);
    line (250, 220, 480, 220);
    floodfill (252, 162, 15);
     setfillstyle (SOLID_FILL, WHITE);
       line (480, 160, 650, 160);
       line (480, 220, 650, 220);
       floodfill (482, 162, 15);
      setfillstyle (SOLID_FILL, GREEN);
       line (250, 220, 650, 220);
       floodfill (252, 278, 15);
        getch();
       closegraph();
```

Output :-

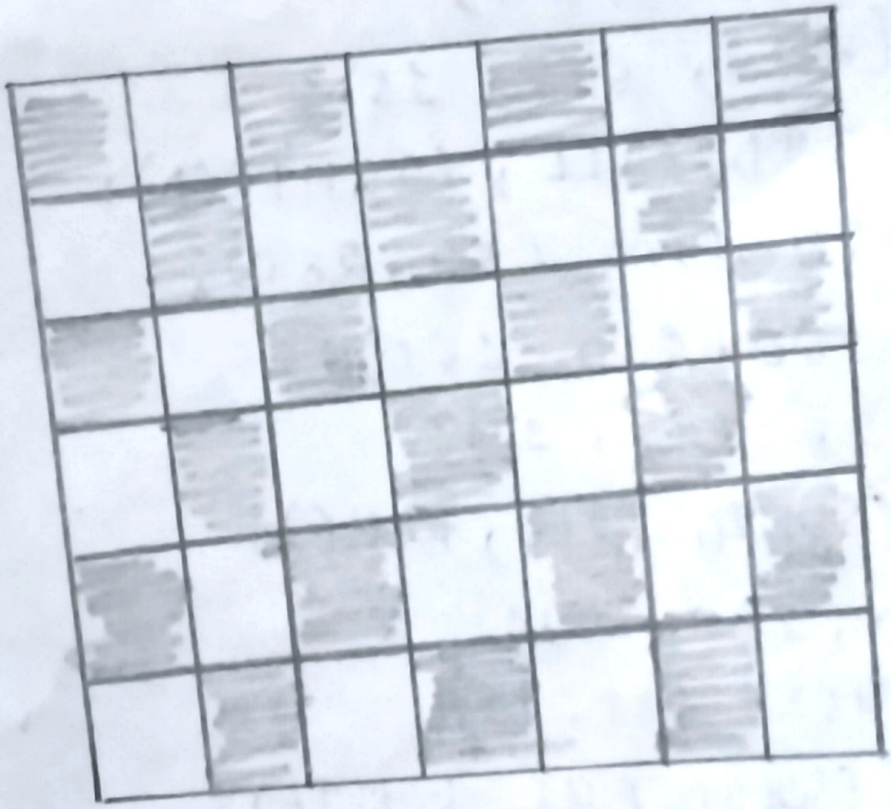


(33) WAP in C to create a chess board in computer graphics.

(22)
```
# include < conio.h>
# include < graphics.h>
# include < dos.h>
# include < stdio.h>
void main()
{
int gr = DETECT, gm;
int r, c, x = 30, y = 30, black = 0;
initgraph (& gr, & gm, "C:\\TURBOC3\\BGI");
for( r = 0; r < 8; r++)
   {
   for( c = 0; c <= 8; c++)
      {
      if (black == 0)
         {
         setcolor( WHITE);
         setfillstyle (SOLID_FILL, BLACK);
      rectangle (x, y, x + 30, y + 30);
   floodfill (x+1, y+1, WHITE);
      black = 1;
      }
      else {
            setcolor (WHITE);
         setfillstyle (SOLID_FILL, WHITE);
         rectangle (x, y, x+30, y+30);
         floodfill (x+1, y+1, WHITE);
               black = 0;
               }
```

```
x = x + 30;
delay(30);
}
if (black == 0)
    black = 1;
else
    black = 0;
    delay(30);
    x = 30;
    y = 30 + y;
}
getch();
closegraph();
}
```

Output :-

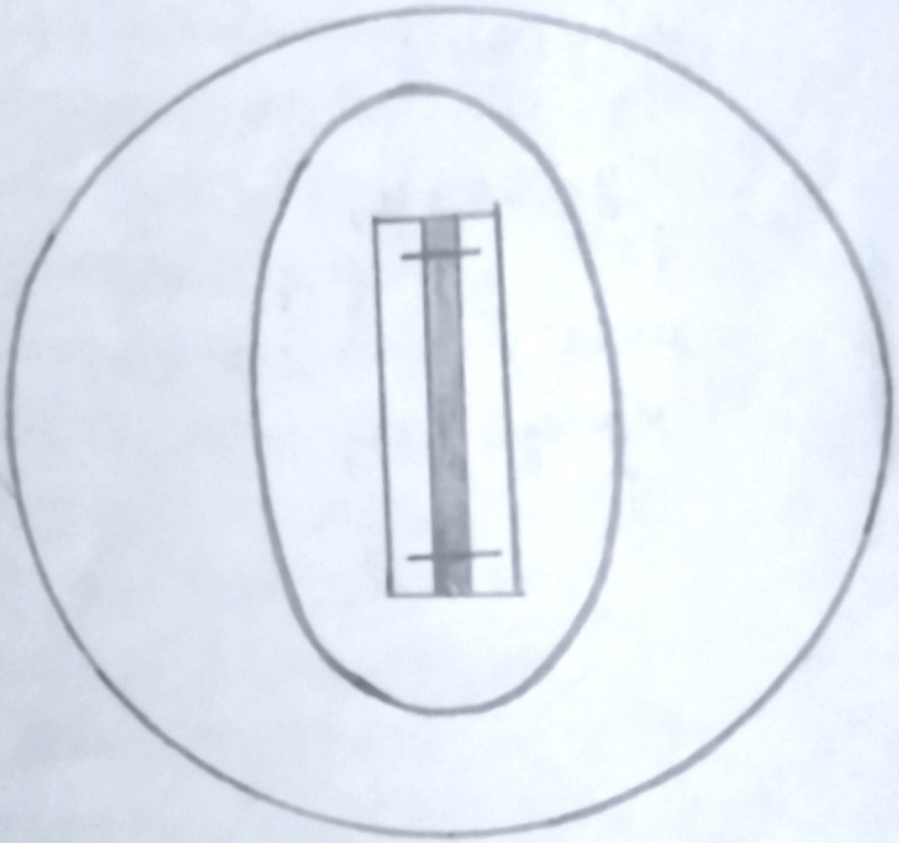


(34) WAP to draw a cricket ground using computer graphics.

(34)

```
# include<conio.h>
# include<graphics.h>
# include<stdio.h>
void main()
{
int gd = DETECT, gm;
initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
circle(700, 350, 300);
setfillstyle(SOLID-FILL, GREEN);
floodfill(402, 350, 15);
ellipse(700, 350, 0, 360, 150, 200);
rectangle(675, 265, 725, 435);
rectangle(690, 265, 710, 435);
setfillstyle(SOLID-FILL, BROWN);
floodfill(695, 300, 15);
rectangle(690, 265, 710, 280);
line(680, 280, 720, 280);
rectangle(690, 435, 710, 420);
line(680, 420, 720, 420);
getch();
closegraph();
}
```

Output :-



(35) WAP in C to create a pendulum clock using computer graphics.

(25)
```
# Include <graphics.h>
# Include <conio.h>
# Include <stdio.h>
  void main()
    {
     int gd = DETECT, gm;
     initgraph (&gd, &gm, "C:\\TURBOC3\\BGI");
rectangle (500, 50, 800, 650);
rectangle (520, 70, 780, 630);
 setfillstyle (SOLID_FILL, BROWN);
   floodfill (505, 55, 15);
      circle (650, 200, 130);
      circle (650, 200, 3);
       setfillstyle (SOLID_FILL, DARKGRAY);
         floodfill (525, 355, 15);
           floodfill (522, 72, 15);
        settextstyle (6, 0, 3);
       outtextxy (697, 100, "01");
       outtextxy (730, 140, "02");
       outtextxy (742, 190, "03");
        outtextxy (721, 240, "04");
        outtextxy (690, 280, "05");
         outtextxy (630, 300, "06");
          outtextxy (578, 280, "07");
          outtextxy (540, 240, "08");
           outtextxy (530, 190, "09");
           outtextxy (537, 140, "10");
           outtextxy (569, 100, "11");
            outtextxy (630, 80, "12");
           line (645, 328, 645, 528);
           line (655, 328, 655, 528);
```

```
circle (650, 546, 20);
setfillstyle (SOLID_FILL, BLACK);
 floodfill (652, 544, 15);
 floodfill (647, 330, 15);
    setcolor (BLUE);
  line (647, 197, 600, 170);
    setcolor (YELLOW);
   line (653, 200, 730, 170);
    setcolor (RED);
   line (650, 203, 630, 290);
    getch();
    closegraph();
     }
```

Output :-