COMPUTER GRAPHICS (ITC09)

NETAJI SUBHASH UNIVERSITY OF TECHNOLOGY

Name: - Arpit Kapoor

Roll: - 2017UIT2581

Q2. Write a Program for 2D line drawing as Raster Graphics Display.

CODE:

```
#include<stdio.h>
#include<graphics.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 (X,Y,RED);  // put pixel at (X,Y)
        X += Xinc;           // increment in x at each step
        Y += Yinc;           // increment in y at each step
        delay(100);          // for visualization of line-
                             // generation step by step
    }
}

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

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

    int X0 = 2, Y0 = 2, X1 = 14, Y1 = 16;
    DDA(2, 2, 14, 16);
    return 0;
}
```

Q3. Write a Program for circle drawing as Raster Graphics Display.

CODE:

```
#include<iostream>
#include <dos.h>
#include <graphics.h>
#include<windows.h>
using namespace std;

void drawCircle(int xc, int yc, int x, int y,int s)
{
    putpixel(xc+x, yc+y, s);
    putpixel(xc-x, yc+y, s);
    putpixel(xc+x, yc-y, s);
    putpixel(xc-x, yc-y, s);
    putpixel(xc+y, yc+x, s);
    putpixel(xc-y, yc+x, s);
    putpixel(xc+y, yc-x, s);
    putpixel(xc-y, yc-x, s);
}

void circleBres(int xc, int yc, int r)
{
    int rr=r;
    do{

    int x = 0, y = rr-1;
    int d = 3 - 2 * (rr-1);
    while (y >= x)
    {

        drawCircle(xc, yc, x, y, 0);
        x++;
        if (d > 0)
        {
            y--;
            d = d + 4 * (x - y) + 10;
        }
        else
            d = d + 4 * x + 6;
        //drawCircle(xc, yc, x, y);


    }
    x = 0, y = rr;
    d = 3 - 2 * rr;
    while (y >= x)
    {

        drawCircle(xc, yc, x, y, 2);
        x++;
        if (d > 0)
```

```
        {
            y--;
            d = d + 4 * (x - y) + 10;
        }
        else
            d = d + 4 * x + 6;
        //drawCircle(xc, yc, x, y);


    }
    delay(100);
    rr+=1;
    }while(rr<=(2*r));

}

int main()
{
    int xc , yc, r;
    int gd = DETECT, gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    cin>>xc>>yc>>r;
    circleBres(xc, yc, r);
    getch();
    closegraph();
}
```

Q4. Write a Program to draw an ellipse using mid point algorithm.

CODE:

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void drawellipse(double x, double y, double xc, double yc) {
    putpixel(xc + x, yc + y, 15);
    putpixel(xc + x, yc - y, 15);
    putpixel(xc - x, yc + y, 15);
    putpixel(xc - x, yc - y, 15);
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    double xc, yc;
    cout << "Enter Center(x,y)=";
    cin >> xc >> yc;
    cout << "Enter a and b = ";
    double rx, ry;
    cin >> rx >> ry;
    double x = 0, y = ry;
    double po = pow(ry, 2) - (pow(rx, 2) * ry) + (pow(rx, 2));
    while (rx * rx * y >= ry * ry * x) {
        if (po <= 0) {
            y = y;
            po = po + 2 * ry * ry * (x + 1) + ry * ry;
        }
        else {
            y = y - 1;
            po = po + 2 * ry * ry * (x + 1) + ry * ry - 2 * rx * rx * (y - 1);
        }
        x = x + 1;
        drawellipse(x, y, xc, yc);
    }
    po = ((x + (1 / 2.0)) * (x + (1 / 2.0)) * ry * ry + (y - 1) * (y - 1) * rx *
rx) - rx * rx * ry * ry;
    while (y != 0) {
        if (po < 0) {
            x = x + 1;
            po = po - 2 * rx * rx * (y - 1) + rx * rx + 2 * ry * ry * (x + 1);
        }
        else {
            x = x;
            po = po - 2 * rx * rx * (y - 1) + rx * rx;
        }
        y = y - 1;
        drawellipse(x, y, xc, yc);
    }
    getch();
}
```

Q5. Write a program to draw circle using Mid Point Algorithm.

CODE:

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void drawcircle(int X, int Y, int radius) {
    int x = radius;
    int y = 0;
    int pk = 0;
    while (x >= y) {
        putpixel(X + x, Y + y, 7);
        putpixel(X + y, Y + x, 7);
        putpixel(X - y, Y + x, 7);
        putpixel(X - x, Y + y, 7);
        putpixel(X - x, Y - y, 7);
        putpixel(X - y, Y - x, 7);
        putpixel(X + y, Y - x, 7);
        putpixel(X + x, Y - y, 7);
        if (pk <= 0) {
            y += 1;
            pk += 2 * y + 1;
        }
        if (pk > 0) {
            x -= 1;
            pk -= 2 * x + 1;
        }
    }
}
int main() {
    int x, y, r;
    cout << "Enter co-ordinates of center : ";
    cin >> x >> y;
    cout << "Enter radius of Circle: ";
    cin >> r;
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    drawcircle(x, y, r);
    getch();
    return 0;
}
```

Q6,7,8, 9,10: Perform All 2D transformations:

CODE:

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

double x[100], y[100];
void multiplyMatrix(double a[3][3], int r1, int c1, double b[3][16], int r2, int
c2, double c[3][16]) {
    if (c1 != r2) {
        cout << "Error in multiply matrix" << endl;
        return;
    }
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            c[i][j] = 0;
            for (int k = 0; k < r2; k++) {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
}
void translate(double matrix[3][16], double dx, double dy, double result[3][16])
{
    double transform[3][3] = {
        {1, 0, dx},
        {0, 1, dy},
        {0, 0, 1}
    };
    multiplyMatrix(transform, 3, 3, matrix, 3, 16, result);
}
void scale(double matrix[3][16], double sx, double sy, double result[3][16]) {
    double transform[3][3] = {
        {sx, 0, 0},
        {0, sy, 0},
        {0, 0, 1}
    };
    multiplyMatrix(transform, 3, 3, matrix, 3, 16, result);
}

void reflect(double matrix [3][16],double result[3][16])
{

    double transform[3][3]={
        {1,0,0},
        {0,-1,0},
        {0,0,1}
    };
    multiplyMatrix(transform,3,3,matrix,3,16,result);
}

```

```
void rotate(double matrix[3][16], double angleRad, double result[3][16]) {
    double Cos = cos(angleRad);
    double Sin = sin(angleRad);
    double transform[3][3] = {
        {Cos, -Sin, 0},
        {Sin, Cos, 0},
        {0, 0, 1}
    };
    multiplyMatrix(transform, 3, 3, matrix, 3, 16, result);
}
void draw(int n) {
    for (int i = 0; i < n; i++) {
        int x1 = x[i];
        int y1 = y[i];
        int x2 = x[(i + 1) % n];
        int y2 = y[(i + 1) % n];
        line(x1, y1, x2, y2);
    }
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    cout << "Number of Vertices : ";
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter coordinates : ";
        cin >> x[i] >> y[i];
    }
    setcolor(WHITE);
    draw(n);
    delay(1000);
    cout << "Available choices : " << endl << "a -> Translation" << endl << "b ->
Scaling" << endl << "c -> Rotation" <<endl<<"d -> Reflection" << endl << "-1 ->
Exit" << endl;
    char c;
    while (true) {
        double sx, sy;
        double matrix[3][16], result[3][16];
        cout << "Choice : ";
        cin >> c;
        if (c == 'a') {
            cout << "Translation Factor x and y: ";
            cin >> sx >> sy;
            for (int i = 0; i < n; i++)
            {
                matrix[0][i] = x[i];
                matrix[1][i] = y[i];
                matrix[2][i] = 1;
            }
            translate(matrix, sx, sy, result);
            for (int i = 0; i < n; i++)
            {
                x[i] = result[0][i];
```

```
                y[i] = result[1][i];

            }
            setcolor(RED);
            draw(n);
            delay(1000);
        }
        else if (c == 'b') {
            cout << "Scaling Factor x and y : ";
            cin >> sx >> sy;
            for (int i = 0; i < n; i++) {
                matrix[0][i] = x[i];
                matrix[1][i] = y[i];
                matrix[2][i] = 1;
            }
            scale(matrix, sx, sy, result);
            for (int i = 0; i < n; i++) {
                x[i] = result[0][i];
                y[i] = result[1][i];
            }
            setcolor(BLUE);
            draw(n);
            delay(1000);
        }
        else if (c == 'c') {
            cout << "Enter Angle : ";
            double angle;
            cin >> angle;
            angle = (3.1428 * angle) / 180.0;
            for (int i = 0; i < n; i++) {
                matrix[0][i] = x[i];
                matrix[1][i] = y[i];
                matrix[2][i] = 1;
            }
            rotate(matrix, angle, result);
            for (int i = 0; i < n; i++) {
                x[i] = result[0][i];
                y[i] = result[1][i];

            }
            setcolor(YELLOW);
            draw(n);
            delay(1000);
        }
        else if(c=='d')
        {
            for (int i = 0; i < n; i++)
            {
                matrix[0][i] = x[i];
                matrix[1][i] = y[i];
                matrix[2][i] = 1;
            }
            reflect(matrix,result);
            for (int i = 0; i < n; i++)
```

```
154.                {
155.                    x[i] = result[0][i];
156.                    y[i] = result[1][i];
157.
158.                }
159.                setcolor(RED);
160.                draw(n);
161.                delay(1000);
162.            }
163.            else break;
164.        }
165.        delay(1000);
166.    }
```

Q12. Write a program for line clipping.

CODE: Cohen Sutherland

```
#include <iostream>
#include<graphics.h>
using namespace std;

// Defining region codes
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 = 400;
const int y_max = 320;
const int x_min = 160;
const int y_min = 160;

int computeCode(double x, double y)
{
    int code = INSIDE;

    if (x < x_min)
        code |= LEFT;
    else if (x > x_max)
        code |= RIGHT;
    if (y < y_min)
        code |= BOTTOM;
    else if (y > y_max)
        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;
            setcolor(GREEN);
            line(x1,y1,x2,y2);
        }
        else
            cout << "Line rejected" << endl;
    }

    int main()
    {
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\TC\\BGI");
        setcolor(RED);
        rectangle(x_min,y_min,x_max,y_max);
        setcolor(WHITE);
        line(280,360,440,160);
        line(40,40,160,40);
        line(200,200,280,280);
        cohenSutherlandClip(200, 200, 280, 280);
        cohenSutherlandClip(280, 360, 440, 160);
        cohenSutherlandClip(40, 40, 160, 40);
        getch();
        closegraph();
        return 0;
    }
```

CODE: LIANG BARSKY

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

int main()
{
    int i, gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    int x1, y1, x2, y2, xmin, xmax, ymin, ymax, xx1, xx2, yy1, yy2, dx, dy;
    cout << "Enter Window Coordinates : ";
    cin >> xmin >> ymin >> xmax >> ymax;
    rectangle(xmin, ymin, xmax, ymax);
    double u1, u2, p[4], q[4], temp;
    cout << "Enter Line Coordinates : ";
    cin >> x1 >> y1 >> x2 >> y2;
    dx = x2 - x1;
    dy = y2 - y1;
    line(x1, y1, x2, y2);
    p[0] = -dx;
```

```
    p[1] = dx;
    p[2] = -dy;
    p[3] = dy;

    q[0] = x1 - xmin;
    q[1] = xmax - x1;
    q[2] = y1 - ymin;
    q[3] = ymax - y1;

    for (i = 0; i < 4; i++) {
        if (p[i] == 0) {
            if (q[i] >= 0) {
                if (i < 2) {
                    if (y1 < ymin) {
                        y1 = ymin;
                    }

                    if (y2 > ymax) {
                        y2 = ymax;
                    }
                }

                if (i > 1) {
                    if (x1 < xmin) {
                        x1 = xmin;
                    }

                    if (x2 > xmax) {
                        x2 = xmax;
                    }
                }
            }
        }
    }

    u1 = 0;
    u2 = 1;

    for (i = 0; i < 4; i++) {
        temp = q[i] / p[i];

        if (p[i] < 0) {
            if (u1 <= temp) {
                u1 = temp;
            }
        }
        else {
            if (u2 > temp) {
                u2 = temp;
            }
        }
    }

    if (u1 < u2)
```

```
    {
        xx1 = x1 + u1 * p[1];
        xx2 = x1 + u2 * p[1];
        yy1 = y1 + u1 * p[3];
        yy2 = y1 + u2 * p[3];
        getch();
        cleardevice();
        rectangle(xmin, ymin, xmax, ymax);
        line(xx1, yy1, xx2, yy2);
    }
    else {
        getch();
        cleardevice();
        rectangle(xmin, ymin, xmax, ymax);
    }
    getch();
}
```

CODE: CYRUS BECK

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

struct points {
    double x;
    double y;
};

struct polygon {
    int npoints;
    points v[1000];
};
double dotproduct(points a, points b) {
    return a.x * b.x + a.y * b.y;
}
points getnormal(points a, points b, points boundary) {
    double dx = b.x - a.x;
    double dy = b.y - a.y;
    points N = { -dy , dx};
    points k = {boundary.x - a.x, boundary.y - a.y};
    double dotresult = dotproduct(k, N);
    if (dotresult == 0) {
        cout << "Error--3 Colinear Points Along Polygon" << endl;
        exit(0);
    }
    if (dotresult < 0) {
        N.x *= -1;
        N.y *= -1;
    }
```

```
    return N;
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    cout << "Enter number of Vertices : ";
    polygon poly;
    cin >> poly.npoints;
    double n = poly.npoints;
    for (int i = 0; i < n; i++) {
        cout << "Enter coordinates x and y : ";
        cin >> poly.v[i].x >> poly.v[i].y;
    }
    for (int i = 0; i < n; i++) {
        line(poly.v[i].x, poly.v[i].y, poly.v[(i + 1) % poly.npoints].x,
poly.v[(i + 1) % poly.npoints].y);
    }
    cout << "Enter coordinates of Line (x1,y1) and (x2,y2) : ";
    points Line[2];
    cin >> Line[0].x >> Line[0].y >> Line[1].x >> Line[1].y;
    line(Line[0].x , Line[0].y , Line[1].x , Line[1].y);
    points D = {Line[1].x - Line[0].x, Line[1].y - Line[0].y};
    double tl = 0, tu = 1;
    for (int i = 0; i < n; i++) {
        points W = {Line[0].x - poly.v[i].x, Line[0].y - poly.v[i].y};
        points N = getnormal(poly.v[i], poly.v[(i + 1) % poly.npoints], poly.v[(i
+ 2) % poly.npoints]);
        //points n = N[i];
        double num = dotproduct(W, N);
        double deno = dotproduct(D, N);
        if (deno == 0) {
            if (num < 0) {
                cout << "no";
                exit(0);
            }
            if (num > 0) continue;
        }
        double x = -num / deno;
        if (deno > 0) {
            tl = max(tl, x);
        }
        else tu = min(tu, x);
    }
    if (tl <= tu) {
        double x1 = Line[0].x + (D.x) * tl;
        double y1 = Line[0].y + (D.y) * tl;
        double x2 = Line[0].x + (D.x) * tu;
        double y2 = Line[0].y + (D.y) * tu;
        getch();
        cleardevice();
        for (int i = 0; i < n; i++) {
            line(poly.v[i].x, poly.v[i].y, poly.v[(i + 1) % poly.npoints].x,
poly.v[(i + 1) % poly.npoints].y);
```

```
82.         }
83.         line(x1 , y1 , x2 , y2);
84.     }
85.     else cout << "Parallel Line";
86.     getch();
87. }
```

Q18 Draw A hut with two windows.

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

initgraph(&gd,&gm,"c:\\tc\\bgi");
setcolor(3);
line(100,200,300,200);
line(100,400,300,400);
line(100,400,100,200);
line(300,400,300,200);
line(300,200,500,200);
line(300,400,500,400);
line(500,200,500,400);
line(170,250,270,250);
line(170,200,270,200);
line(170,250,170,400);
line(270,250,270,400);
line(200,50,400,50);
line(400,50,500,200);
line(300,200,200,50);
line(100,200,200,50);
getch();
closegraph();
return 0;
}
```

Q20. Draw a Ceiling Fan

CODE:

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;



int main()
{
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"C:\\TC\\BGI");
    setbkcolor(YELLOW);
    int speed;
    while(1)
    {
        for(int x=0;x<=360;x+=3)
        {
            if(kbhit())
                return 0;
            pieslice(250,250,0,360,35);
            sector(250,250,x,x+15,150,150);
            sector(250,250,x+120,x+135,150,150);
            sector(250,250,x+240,x+255,150,150);
            delay(10);
            cleardevice();
        }
    }
}
```

Q.21 Draw a Full moon and three stars.

CODE:

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;

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

    setcolor(LIGHTGRAY);
    pieslice(100,100,0,360,40);
    line(15,10,10,20);
    line(10,20,20,20);
    line(20,20.0,15,10);
    line(10,12.5,20,12.5);
    line(10,12.5,15,22.5);
    line(15,22.5,20,12.5);

    line(30,10,25,20);
    line(25,20,35,20);
    line(35,20.0,30,10);
    line(25,12.5,35,12.5);
    line(25,12.5,30,22.5);
    line(30,22.5,35,12.5);

    line(20,25,15,35);
    line(15,35,25,35);
    line(25,35.0,20,25);
    line(15,27.5,25,27.5);
    line(15,27.5,20,37.5);
    line(20,37.5,25,27.5);
    getch();
    closegraph();
    return 0;
}
```

Q.24 Draw a bar chart having at least 3 bars.

CODE:

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
int main()
{
   int gd=DETECT,gm;
   initgraph(&gd,&gm,"C:\\TC\\BGI");
   line(20,400,500,400);
   line(20,400,20,40);
   for(int i=400;i>=300;i--)
   {
       setcolor(YELLOW);
       //setfillstyle(SOLID_FILL,YELLOW);
       rectangle(50,i,90,400);
       delay(10);
   }
   for(int i=400;i>=200;i--)
   {
       setcolor(RED);
       //setfillstyle(SOLID_FILL,RED);
       rectangle(120,i,160,400);
       delay(10);
   }
      for(int i=400;i>=250;i--)
   {
       setcolor(BLUE);
       //setfillstyle(SOLID_FILL,BLUE);
       rectangle(190,i,230,400);
       delay(10);
   }
      for(int i=400;i>=100;i--)
   {
       setcolor(GREEN);
       rectangle(260,i,300,400);
       delay(10);
   }
     for(int i=400;i>=70;i--)
   {
       setcolor(CYAN);
       rectangle(330,i,370,400);
       delay(10);
   }
   getch();
   closegraph();
   return 0;
}
```

Q.25 Draw a rotating coin on a table

CODE:

```
#include<bits/stdc++.h>
#include<graphics.h>
int main()
{
  int gd=DETECT,gm;
  initgraph(&gd,&gm,"C:\\TC\\BGI");
  while(1)
  {
      int a=15, b=15;
      for(;a>0;a--)
      {
          setbkcolor(BLUE);
          fillellipse(300,400,a,b);
          line(250,375,350,375);
          line(225,425,325,425);
          line(250,375,225,425);
          line(350,375,325,425);
          line(225,425,225,600);
          line(325,425,325,600);
          line(350,375,350,450);
          delay(100);
          cleardevice();
      }
      for(;a<=15;a++)
      {
          fillellipse(300,400,a,b);
          line(250,375,350,375);
          line(225,425,325,425);
          line(250,375,225,425);
          line(350,375,325,425);
          line(225,425,225,600);
          line(325,425,325,600);
          line(350,375,350,450);
          delay(100);
          cleardevice();
      }
  }
}
```

Q26 Write your name in Hindi using Bezier Curve.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <math.h>
#include<iostream>
using namespace std;
void bezier (int x[4], int y[4])
{
   int i;
   double t;
   for (t = 0.0; t < 1.0; t += 0.0005)
   {
   double x_ = pow (1-t, 3) * x[0] + 3 * t * pow (1-t, 2) * x[1] +
         3 * pow (t, 2) * (1-t) * x[2] + pow (t, 3) * x[3];

   double y_ = pow (1-t, 3) * y[0] + 3 * t * pow (1-t, 2) * y[1] +
         3 * pow (t, 2) * (1-t) * y[2] + pow (t, 3) * y[3];

   putpixel (x_, y_, WHITE);
   }

   return;
}

int main()
{
   int gd = DETECT, gm;
   initgraph (&gd, &gm, "..\\bgi");
   line(10,100,500,100);
   line(150,100,150,300);
   int x[]={50,120,120,35};
   int y[]={100,100,200,200};
   bezier (x, y);
   line(50,200,150,200);
   line(290,100,290,300);
   line(170,100,170,300);
   int a[]={50,120,120,10};
   int b[]={200,200,300,300};
   bezier (a,b);
   int c[]={180,200,260,290};
   int d[]={100,250,250,100};
   bezier (c,d);
   int m[]={170,170,220,290};
   int n[]={100,20,60,100};
   bezier (m,n);
   line(420,100,420,300);
   line(420,200,360,200);
   int p[]={360,340,350,360};
   int q[]={200,200,300,300};
```

```
    bezier(p,q);
    int j[]={290,290,305,320};
    int k[]={100,50,40,40};
    bezier (j,k);
    getch();
    closegraph();
    return 0;
}
```

Q.28 Draw a Bouncing Ball

CODE:

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

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

  int toggle=0;
  while(1){
  for(int y=30;y<=400;y+=2)
  {
      pieslice(300,y,0,360,30);
      line(0,430,1000,430);
      delay(3);
      cleardevice();
  }
  for(int f=400;f>=30;f-=2)
  {
      pieslice(300,f,0,360,30);
      line(0,430,1000,430);
      delay(3);
      cleardevice();
  //    delay(30);
  }
    }
  outtext("press any key to exit");
  getch();
  closegraph();

  return 0;

}
```