

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:

```
1. #include<stdio.h>
2. #include<graphics.h>
3.
4. int abs (int n)
5. {
6.     return ( (n>0) ? n : ( n * (-1) ));
7. }
8.
9. void DDA(int X0, int Y0, int X1, int Y1)
10. {
11.     int dx = X1 - X0;
12.     int dy = Y1 - Y0;
13.
14.     int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
15.
16.     float Xinc = dx / (float) steps;
17.     float Yinc = dy / (float) steps;
18.
19.     float X = X0;
20.     float Y = Y0;
21.     for (int i = 0; i <= steps; i++)
22.     {
23.         putpixel (X,Y,RED); // put pixel at (X,Y)
24.         X += Xinc;          // increment in x at each step
25.         Y += Yinc;          // increment in y at each step
26.         delay(100);         // for visualization of line-
27.                             // generation step by step
28.     }
29. }
30.
31. // Driver program
32. int main()
33. {
34.     int gd = DETECT, gm;
35.
36.     // Initialize graphics function
37.     initgraph (&gd, &gm, "");
38.
39.     int X0 = 2, Y0 = 2, X1 = 14, Y1 = 16;
40.     DDA(2, 2, 14, 16);
41.     return 0;
42. }
```

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

CODE:

```
1. #include<iostream>
2. #include <dos.h>
3. #include <graphics.h>
4. #include<windows.h>
5. using namespace std;
6.
7. void drawCircle(int xc, int yc, int x, int y,int s)
8. {
9.     putpixel(xc+x, yc+y, s);
10.    putpixel(xc-x, yc+y, s);
11.    putpixel(xc+x, yc-y, s);
12.    putpixel(xc-x, yc-y, s);
13.    putpixel(xc+y, yc+x, s);
14.    putpixel(xc-y, yc+x, s);
15.    putpixel(xc+y, yc-x, s);
16.    putpixel(xc-y, yc-x, s);
17. }
18.
19. void circleBres(int xc, int yc, int r)
20. {
21.     int rr=r;
22.     do{
23.
24.         int x = 0, y = rr-1;
25.         int d = 3 - 2 * (rr-1);
26.         while (y >= x)
27.         {
28.
29.             drawCircle(xc, yc, x, y, 0);
30.             x++;
31.             if (d > 0)
32.             {
33.                 y--;
34.                 d = d + 4 * (x - y) + 10;
35.             }
36.             else
37.                 d = d + 4 * x + 6;
38.             //drawCircle(xc, yc, x, y);
39.
40.
41.         }
42.         x = 0, y = rr;
43.         d = 3 - 2 * rr;
44.         while (y >= x)
45.         {
46.
47.             drawCircle(xc, yc, x, y, 2);
48.             x++;
49.             if (d > 0)
```

```

50.     {
51.         y--;
52.         d = d + 4 * (x - y) + 10;
53.     }
54.     else
55.         d = d + 4 * x + 6;
56.     //drawCircle(xc, yc, x, y);
57.
58.
59.     }
60.     delay(100);
61.     rr+=1;
62.     }while(rr<=(2*r));
63.
64. }
65.
66. int main()
67. {
68.     int xc , yc, r;
69.     int gd = DETECT, gm;
70.     initgraph(&gd,&gm,"C:\\TC\\BGI");
71.     cin>>xc>>yc>>r;
72.     circleBres(xc, yc, r);
73.     getch();
74.     closegraph();
75. }

```

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

CODE:

```
1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. using namespace std;
4. void drawellipse(double x, double y, double xc, double yc) {
5.     putpixel(xc + x, yc + y, 15);
6.     putpixel(xc + x, yc - y, 15);
7.     putpixel(xc - x, yc + y, 15);
8.     putpixel(xc - x, yc - y, 15);
9. }
10. int main() {
11.     int gd = DETECT, gm;
12.     initgraph(&gd, &gm, "C:\\\\TC\\\\BGI");
13.     double xc, yc;
14.     cout << "Enter Center(x,y)=";
15.     cin >> xc >> yc;
16.     cout << "Enter a and b = ";
17.     double rx, ry;
18.     cin >> rx >> ry;
19.     double x = 0, y = ry;
20.     double po = pow(ry, 2) - (pow(rx, 2) * ry) + (pow(rx, 2));
21.     while (rx * rx * y >= ry * ry * x) {
22.         if (po <= 0) {
23.             y = y;
24.             po = po + 2 * ry * ry * (x + 1) + ry * ry;
25.         }
26.         else {
27.             y = y - 1;
28.             po = po + 2 * ry * ry * (x + 1) + ry * ry - 2 * rx * rx * (y - 1);
29.         }
30.         x = x + 1;
31.         drawellipse(x, y, xc, yc);
32.     }
33.     po = ((x + (1 / 2.0)) * (x + (1 / 2.0)) * ry * ry + (y - 1) * (y - 1) * rx *
rx) - rx * rx * ry * ry;
34.     while (y != 0) {
35.         if (po < 0) {
36.             x = x + 1;
37.             po = po - 2 * rx * rx * (y - 1) + rx * rx + 2 * ry * ry * (x + 1);
38.         }
39.         else {
40.             x = x;
41.             po = po - 2 * rx * rx * (y - 1) + rx * rx;
42.         }
43.         y = y - 1;
44.         drawellipse(x, y, xc, yc);
45.     }
46.     getch();
47. }
```

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

CODE:

```
1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. using namespace std;
4. void drawcircle(int X, int Y, int radius) {
5.     int x = radius;
6.     int y = 0;
7.     int pk = 0;
8.     while (x >= y) {
9.         putpixel(X + x, Y + y, 7);
10.        putpixel(X + y, Y + x, 7);
11.        putpixel(X - y, Y + x, 7);
12.        putpixel(X - x, Y + y, 7);
13.        putpixel(X - x, Y - y, 7);
14.        putpixel(X - y, Y - x, 7);
15.        putpixel(X + y, Y - x, 7);
16.        putpixel(X + x, Y - y, 7);
17.        if (pk <= 0) {
18.            y += 1;
19.            pk += 2 * y + 1;
20.        }
21.        if (pk > 0) {
22.            x -= 1;
23.            pk -= 2 * x + 1;
24.        }
25.    }
26. }
27. int main() {
28.     int x, y, r;
29.     cout << "Enter co-ordinates of center : ";
30.     cin >> x >> y;
31.     cout << "Enter radius of Circle: ";
32.     cin >> r;
33.     int gd = DETECT, gm;
34.     initgraph(&gd, &gm, "C:\\TC\\BGI");
35.     drawcircle(x, y, r);
36.     getch();
37.     return 0;
38. }
```

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

CODE:

```
1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. using namespace std;
4.
5. double x[100], y[100];
6. void multiplyMatrix(double a[3][3], int r1, int c1, double b[3][16], int r2, int
   c2, double c[3][16]) {
7.     if (c1 != r2) {
8.         cout << "Error in multiply matrix" << endl;
9.         return;
10.    }
11.    for (int i = 0; i < r1; i++) {
12.        for (int j = 0; j < c2; j++) {
13.            c[i][j] = 0;
14.            for (int k = 0; k < r2; k++) {
15.                c[i][j] += a[i][k] * b[k][j];
16.            }
17.        }
18.    }
19. }
20. void translate(double matrix[3][16], double dx, double dy, double result[3][16])
   {
21.     double transform[3][3] = {
22.         {1, 0, dx},
23.         {0, 1, dy},
24.         {0, 0, 1}
25.     };
26.     multiplyMatrix(transform, 3, 3, matrix, 3, 16, result);
27. }
28. void scale(double matrix[3][16], double sx, double sy, double result[3][16]) {
29.     double transform[3][3] = {
30.         {sx, 0, 0},
31.         {0, sy, 0},
32.         {0, 0, 1}
33.     };
34.     multiplyMatrix(transform, 3, 3, matrix, 3, 16, result);
35. }
36.
37. void reflect(double matrix [3][16],double result[3][16])
38. {
39.
40.     double transform[3][3]={
41.         {1,0,0},
42.         {0,-1,0},
43.         {0,0,1}
44.     };
45.     multiplyMatrix(transform,3,3,matrix,3,16,result);
46. }
47.
```

```

48. void rotate(double matrix[3][16], double angleRad, double result[3][16]) {
49.     double Cos = cos(angleRad);
50.     double Sin = sin(angleRad);
51.     double transform[3][3] = {
52.         {Cos, -Sin, 0},
53.         {Sin, Cos, 0},
54.         {0, 0, 1}
55.     };
56.     multiplyMatrix(transform, 3, 3, matrix, 3, 16, result);
57. }
58. void draw(int n) {
59.     for (int i = 0; i < n; i++) {
60.         int x1 = x[i];
61.         int y1 = y[i];
62.         int x2 = x[(i + 1) % n];
63.         int y2 = y[(i + 1) % n];
64.         line(x1, y1, x2, y2);
65.     }
66. }
67. int main() {
68.     int gd = DETECT, gm;
69.     initgraph(&gd, &gm, "C:\\\\TC\\\\BGI");
70.     cout << "Number of Vertices : ";
71.     int n;
72.     cin >> n;
73.     for (int i = 0; i < n; i++) {
74.         cout << "Enter coordinates : ";
75.         cin >> x[i] >> y[i];
76.     }
77.     setcolor(WHITE);
78.     draw(n);
79.     delay(1000);
80.     cout << "Available choices : " << endl << "a -> Translation" << endl << "b ->
Scaling" << endl << "c -> Rotation" << endl << "d -> Reflection" << endl << "-1 ->
Exit" << endl;
81.     char c;
82.     while (true) {
83.         double sx, sy;
84.         double matrix[3][16], result[3][16];
85.         cout << "Choice : ";
86.         cin >> c;
87.         if (c == 'a') {
88.             cout << "Translation Factor x and y: ";
89.             cin >> sx >> sy;
90.             for (int i = 0; i < n; i++)
91.             {
92.                 matrix[0][i] = x[i];
93.                 matrix[1][i] = y[i];
94.                 matrix[2][i] = 1;
95.             }
96.             translate(matrix, sx, sy, result);
97.             for (int i = 0; i < n; i++)
98.             {
99.                 x[i] = result[0][i];

```



```

100.         y[i] = result[1][i];
101.
102.     }
103.     setcolor(RED);
104.     draw(n);
105.     delay(1000);
106. }
107. else if (c == 'b') {
108.     cout << "Scaling Factor x and y : ";
109.     cin >> sx >> sy;
110.     for (int i = 0; i < n; i++) {
111.         matrix[0][i] = x[i];
112.         matrix[1][i] = y[i];
113.         matrix[2][i] = 1;
114.     }
115.     scale(matrix, sx, sy, result);
116.     for (int i = 0; i < n; i++) {
117.         x[i] = result[0][i];
118.         y[i] = result[1][i];
119.     }
120.     setcolor(BLUE);
121.     draw(n);
122.     delay(1000);
123. }
124. else if (c == 'c') {
125.     cout << "Enter Angle : ";
126.     double angle;
127.     cin >> angle;
128.     angle = (3.1428 * angle) / 180.0;
129.     for (int i = 0; i < n; i++) {
130.         matrix[0][i] = x[i];
131.         matrix[1][i] = y[i];
132.         matrix[2][i] = 1;
133.     }
134.     rotate(matrix, angle, result);
135.     for (int i = 0; i < n; i++) {
136.         x[i] = result[0][i];
137.         y[i] = result[1][i];
138.     }
139.     setcolor(YELLOW);
140.     draw(n);
141.     delay(1000);
142. }
143. else if(c=='d')
144. {
145.     for (int i = 0; i < n; i++)
146.     {
147.         matrix[0][i] = x[i];
148.         matrix[1][i] = y[i];
149.         matrix[2][i] = 1;
150.     }
151.     reflect(matrix,result);
152.     for (int i = 0; i < n; i++)
153.

```

```
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

```
1. #include <iostream>
2. #include<graphics.h>
3. using namespace std;
4.
5. // Defining region codes
6. const int INSIDE = 0; // 0000
7. const int LEFT = 1; // 0001
8. const int RIGHT = 2; // 0010
9. const int BOTTOM = 4; // 0100
10. const int TOP = 8; // 1000
11.
12. const int x_max = 400;
13. const int y_max = 320;
14. const int x_min = 160;
15. const int y_min = 160;
16.
17. int computeCode(double x, double y)
18. {
19.     int code = INSIDE;
20.
21.     if (x < x_min)
22.         code |= LEFT;
23.     else if (x > x_max)
24.         code |= RIGHT;
25.     if (y < y_min)
26.         code |= BOTTOM;
27.     else if (y > y_max)
28.         code |= TOP;
29.     return code;
30. }
31.
32. void cohenSutherlandClip(double x1, double y1,
33.                          double x2, double y2)
34. {
35.     int code1 = computeCode(x1, y1);
36.     int code2 = computeCode(x2, y2);
37.
38.
39.     bool accept = false;
40.
41.     while (true)
42.     {
43.         if ((code1 == 0) && (code2 == 0))
44.         {
45.
46.             accept = true;
47.             break;
48.         }
49.         else if (code1 & code2)
```

```

50.     {
51.
52.         break;
53.     }
54.     else
55.     {
56.
57.         int code_out;
58.         double x, y;
59.
60.
61.         if (code1 != 0)
62.             code_out = code1;
63.         else
64.             code_out = code2;
65.
66.
67.         if (code_out & TOP)
68.         {
69.
70.             x = x1 + (x2 - x1) * (y_max - y1) / (y2 - y1);
71.             y = y_max;
72.         }
73.         else if (code_out & BOTTOM)
74.         {
75.
76.             x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1);
77.             y = y_min;
78.         }
79.         else if (code_out & RIGHT)
80.         {
81.
82.             y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1);
83.             x = x_max;
84.         }
85.         else if (code_out & LEFT)
86.         {
87.
88.             y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1);
89.             x = x_min;
90.         }
91.
92.         if (code_out == code1)
93.         {
94.             x1 = x;
95.             y1 = y;
96.             code1 = computeCode(x1, y1);
97.         }
98.         else
99.         {
100.             x2 = x;
101.             y2 = y;
102.             code2 = computeCode(x2, y2);
103.         }

```

```

104.     }
105. }
106. if (accept)
107. {
108.     cout << "Line accepted from " << x1 << ", "
109.         << y1 << " to " << x2 << ", " << y2 << endl;
110.     setcolor(GREEN);
111.     line(x1,y1,x2,y2);
112. }
113. else
114.     cout << "Line rejected" << endl;
115. }
116.
117. int main()
118. {
119.     int gd = DETECT, gm;
120.     initgraph(&gd, &gm, "C:\\TC\\BGI");
121.     setcolor(RED);
122.     rectangle(x_min,y_min,x_max,y_max);
123.     setcolor(WHITE);
124.     line(280,360,440,160);
125.     line(40,40,160,40);
126.     line(200,200,280,280);
127.     cohenSutherlandClip(200, 200, 280, 280);
128.     cohenSutherlandClip(280, 360, 440, 160);
129.     cohenSutherlandClip(40, 40, 160, 40);
130.     getch();
131.     closegraph();
132.     return 0;
133. }

```

CODE: LIANG BARSKY

```

1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. using namespace std;
4.
5. int main()
6. {
7.     int i, gd = DETECT, gm;
8.     initgraph(&gd, &gm, "C:\\TC\\BGI");
9.     int x1, y1, x2, y2, xmin, xmax, ymin, ymax, xx1, xx2, yy1, yy2, dx, dy;
10.    cout << "Enter Window Coordinates : ";
11.    cin >> xmin >> ymin >> xmax >> ymax;
12.    rectangle(xmin, ymin, xmax, ymax);
13.    double u1, u2, p[4], q[4], temp;
14.    cout << "Enter Line Coordinates : ";
15.    cin >> x1 >> y1 >> x2 >> y2;
16.    dx = x2 - x1;
17.    dy = y2 - y1;
18.    line(x1, y1, x2, y2);
19.    p[0] = -dx;

```

```

20.     p[1] = dx;
21.     p[2] = -dy;
22.     p[3] = dy;
23.
24.     q[0] = x1 - xmin;
25.     q[1] = xmax - x1;
26.     q[2] = y1 - ymin;
27.     q[3] = ymax - y1;
28.
29.     for (i = 0; i < 4; i++) {
30.         if (p[i] == 0) {
31.             if (q[i] >= 0) {
32.                 if (i < 2) {
33.                     if (y1 < ymin) {
34.                         y1 = ymin;
35.                     }
36.
37.                     if (y2 > ymax) {
38.                         y2 = ymax;
39.                     }
40.                 }
41.
42.                 if (i > 1) {
43.                     if (x1 < xmin) {
44.                         x1 = xmin;
45.                     }
46.
47.                     if (x2 > xmax) {
48.                         x2 = xmax;
49.                     }
50.                 }
51.             }
52.         }
53.     }
54.
55.     u1 = 0;
56.     u2 = 1;
57.
58.     for (i = 0; i < 4; i++) {
59.         temp = q[i] / p[i];
60.
61.         if (p[i] < 0) {
62.             if (u1 <= temp) {
63.                 u1 = temp;
64.             }
65.         }
66.         else {
67.             if (u2 > temp) {
68.                 u2 = temp;
69.             }
70.         }
71.     }
72.
73.     if (u1 < u2)

```

```

74.     {
75.         xx1 = x1 + u1 * p[1];
76.         xx2 = x1 + u2 * p[1];
77.         yy1 = y1 + u1 * p[3];
78.         yy2 = y1 + u2 * p[3];
79.         getch();
80.         cleardevice();
81.         rectangle(xmin, ymin, xmax, ymax);
82.         line(xx1, yy1, xx2, yy2);
83.     }
84.     else {
85.         getch();
86.         cleardevice();
87.         rectangle(xmin, ymin, xmax, ymax);
88.     }
89.     getch();
90. }

```

CODE: CYRUS BECK

```

1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. using namespace std;
4.
5. struct points {
6.     double x;
7.     double y;
8. };
9.
10. struct polygon {
11.     int npoints;
12.     points v[1000];
13. };
14. double dotproduct(points a, points b) {
15.     return a.x * b.x + a.y * b.y;
16. }
17. points getnormal(points a, points b, points boundary) {
18.     double dx = b.x - a.x;
19.     double dy = b.y - a.y;
20.     points N = { -dy , dx};
21.     points k = {boundary.x - a.x, boundary.y - a.y};
22.     double dotresult = dotproduct(k, N);
23.     if (dotresult == 0) {
24.         cout << "Error--3 Colinear Points Along Polygon" << endl;
25.         exit(0);
26.     }
27.     if (dotresult < 0) {
28.         N.x *= -1;
29.         N.y *= -1;
30.     }

```

```

31.     return N;
32. }
33.
34. int main() {
35.     int gd = DETECT, gm;
36.     initgraph(&gd, &gm, "C:\\TC\\BGI");
37.     cout << "Enter number of Vertices : ";
38.     polygon poly;
39.     cin >> poly.npoints;
40.     double n = poly.npoints;
41.     for (int i = 0; i < n; i++) {
42.         cout << "Enter coordinates x and y : ";
43.         cin >> poly.v[i].x >> poly.v[i].y;
44.     }
45.     for (int i = 0; i < n; i++) {
46.         line(poly.v[i].x, poly.v[i].y, poly.v[(i + 1) % poly.npoints].x,
poly.v[(i + 1) % poly.npoints].y);
47.     }
48.     cout << "Enter coordinates of Line (x1,y1) and (x2,y2) : ";
49.     points Line[2];
50.     cin >> Line[0].x >> Line[0].y >> Line[1].x >> Line[1].y;
51.     line(Line[0].x, Line[0].y, Line[1].x, Line[1].y);
52.     points D = {Line[1].x - Line[0].x, Line[1].y - Line[0].y};
53.     double t1 = 0, tu = 1;
54.     for (int i = 0; i < n; i++) {
55.         points W = {Line[0].x - poly.v[i].x, Line[0].y - poly.v[i].y};
56.         points N = getnormal(poly.v[i], poly.v[(i + 1) % poly.npoints], poly.v[(i
+ 2) % poly.npoints]);
57.         //points n = N[i];
58.         double num = dotproduct(W, N);
59.         double deno = dotproduct(D, N);
60.         if (deno == 0) {
61.             if (num < 0) {
62.                 cout << "no";
63.                 exit(0);
64.             }
65.             if (num > 0) continue;
66.         }
67.         double x = -num / deno;
68.         if (deno > 0) {
69.             t1 = max(t1, x);
70.         }
71.         else tu = min(tu, x);
72.     }
73.     if (t1 <= tu) {
74.         double x1 = Line[0].x + (D.x) * t1;
75.         double y1 = Line[0].y + (D.y) * t1;
76.         double x2 = Line[0].x + (D.x) * tu;
77.         double y2 = Line[0].y + (D.y) * tu;
78.         getch();
79.         cleardevice();
80.         for (int i = 0; i < n; i++) {
81.             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.

```
1. #include<stdio.h>
2. #include<conio.h>
3. #include<graphics.h>
4. int main()
5. {
6.     int gd=DETECT, gm;
7.
8.     initgraph(&gd, &gm, "c:\\tc\\bgi");
9.     setcolor(3);
10. line(100,200,300,200);
11. line(100,400,300,400);
12. line(100,400,100,200);
13. line(300,400,300,200);
14. line(300,200,500,200);
15. line(300,400,500,400);
16. line(500,200,500,400);
17. line(170,250,270,250);
18. line(170,200,270,200);
19. line(170,250,170,400);
20. line(270,250,270,400);
21. line(200,50,400,50);
22. line(400,50,500,200);
23. line(300,200,200,50);
24. line(100,200,200,50);
25. getch();
26. closegraph();
27. return 0;
28. }
```

Q20. Draw a Ceiling Fan

CODE:

```
1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. using namespace std;
4.
5.
6.
7. int main()
8. {
9.     int gd=DETECT,gm;
10.    initgraph(&gd,&gm,"C:\\TC\\BGI");
11.    setbkcolor(YELLOW);
12.    int speed;
13.    while(1)
14.    {
15.        for(int x=0;x<=360;x+=3)
16.        {
17.            if(kbhit())
18.                return 0;
19.            pieslice(250,250,0,360,35);
20.            sector(250,250,x,x+15,150,150);
21.            sector(250,250,x+120,x+135,150,150);
22.            sector(250,250,x+240,x+255,150,150);
23.            delay(10);
24.            cleardevice();
25.        }
26.    }
27. }
```

Q.21 Draw a Full moon and three stars.

CODE:

```
1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. using namespace std;
4.
5. int main()
6. {
7.     int gd = DETECT, gm;
8.     initgraph(&gd, &gm, "C:\\\\TC\\\\BGI");
9.
10.    setcolor(LIGHTGRAY);
11.    pieslice(100,100,0,360,40);
12.    line(15,10,10,20);
13.    line(10,20,20,20);
14.    line(20,20.0,15,10);
15.    line(10,12.5,20,12.5);
16.    line(10,12.5,15,22.5);
17.    line(15,22.5,20,12.5);
18.
19.    line(30,10,25,20);
20.    line(25,20,35,20);
21.    line(35,20.0,30,10);
22.    line(25,12.5,35,12.5);
23.    line(25,12.5,30,22.5);
24.    line(30,22.5,35,12.5);
25.
26.    line(20,25,15,35);
27.    line(15,35,25,35);
28.    line(25,35.0,20,25);
29.    line(15,27.5,25,27.5);
30.    line(15,27.5,20,37.5);
31.    line(20,37.5,25,27.5);
32.    getch();
33.    closegraph();
34.    return 0;
35. }
```

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

CODE:

```
1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. using namespace std;
4. int main()
5. {
6.     int gd=DETECT,gm;
7.     initgraph(&gd,&gm,"C:\\TC\\BGI");
8.     line(20,400,500,400);
9.     line(20,400,20,40);
10.    for(int i=400;i>=300;i--)
11.    {
12.        setcolor(YELLOW);
13.        //setfillstyle(SOLID_FILL,YELLOW);
14.        rectangle(50,i,90,400);
15.        delay(10);
16.    }
17.    for(int i=400;i>=200;i--)
18.    {
19.        setcolor(RED);
20.        //setfillstyle(SOLID_FILL,RED);
21.        rectangle(120,i,160,400);
22.        delay(10);
23.    }
24.    for(int i=400;i>=250;i--)
25.    {
26.        setcolor(BLUE);
27.        //setfillstyle(SOLID_FILL,BLUE);
28.        rectangle(190,i,230,400);
29.        delay(10);
30.    }
31.    for(int i=400;i>=100;i--)
32.    {
33.        setcolor(GREEN);
34.        rectangle(260,i,300,400);
35.        delay(10);
36.    }
37.    for(int i=400;i>=70;i--)
38.    {
39.        setcolor(CYAN);
40.        rectangle(330,i,370,400);
41.        delay(10);
42.    }
43.    getch();
44.    closegraph();
45.    return 0;
46. }
```

Q.25 Draw a rotating coin on a table

CODE:

```
1. #include<bits/stdc++.h>
2. #include<graphics.h>
3. int main()
4. {
5.     int gd=DETECT, gm;
6.     initgraph(&gd, &gm, "C:\\TC\\BGI");
7.     while(1)
8.     {
9.         int a=15, b=15;
10.        for(;a>0;a--)
11.        {
12.            setbkcolor(BLUE);
13.            fillellipse(300,400,a,b);
14.            line(250,375,350,375);
15.            line(225,425,325,425);
16.            line(250,375,225,425);
17.            line(350,375,325,425);
18.            line(225,425,225,600);
19.            line(325,425,325,600);
20.            line(350,375,350,450);
21.            delay(100);
22.            cleardevice();
23.        }
24.        for(;a<=15;a++)
25.        {
26.            fillellipse(300,400,a,b);
27.            line(250,375,350,375);
28.            line(225,425,325,425);
29.            line(250,375,225,425);
30.            line(350,375,325,425);
31.            line(225,425,225,600);
32.            line(325,425,325,600);
33.            line(350,375,350,450);
34.            delay(100);
35.            cleardevice();
36.        }
37.    }
38. }
```

Q26 Write your name in Hindi using Bezier Curve.

CODE:

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <graphics.h>
4. #include <math.h>
5. #include <iostream>
6. using namespace std;
7. void bezier (int x[4], int y[4])
8. {
9.     int i;
10.    double t;
11.    for (t = 0.0; t < 1.0; t += 0.0005)
12.    {
13.        double x_ = pow (1-t, 3) * x[0] + 3 * t * pow (1-t, 2) * x[1] +
14.            3 * pow (t, 2) * (1-t) * x[2] + pow (t, 3) * x[3];
15.
16.        double y_ = pow (1-t, 3) * y[0] + 3 * t * pow (1-t, 2) * y[1] +
17.            3 * pow (t, 2) * (1-t) * y[2] + pow (t, 3) * y[3];
18.
19.        putpixel (x_, y_, WHITE);
20.    }
21.
22.    return;
23. }
24.
25. int main()
26. {
27.    int gd = DETECT, gm;
28.    initgraph (&gd, &gm, "..\\bgi");
29.    line(10,100,500,100);
30.    line(150,100,150,300);
31.    int x[]={50,120,120,35};
32.    int y[]={100,100,200,200};
33.    bezier (x, y);
34.    line(50,200,150,200);
35.    line(290,100,290,300);
36.    line(170,100,170,300);
37.    int a[]={50,120,120,10};
38.    int b[]={200,200,300,300};
39.    bezier (a,b);
40.    int c[]={180,200,260,290};
41.    int d[]={100,250,250,100};
42.    bezier (c,d);
43.    int m[]={170,170,220,290};
44.    int n[]={100,20,60,100};
45.    bezier (m,n);
46.    line(420,100,420,300);
47.    line(420,200,360,200);
48.    int p[]={360,340,350,360};
49.    int q[]={200,200,300,300};
```

```
50. bezier(p,q);
51. int j[]={290,290,305,320};
52. int k[]={100,50,40,40};
53. bezier (j,k);
54. getch();
55. closegraph();
56. return 0;
57. }
```


Q.28 Draw a Bouncing Ball

CODE:

```
1. #include<bits/stdc++.h>
2. #include<graphics.h>
3.
4. int main()
5. {
6.     int gd=DETECT,gm;
7.     initgraph(&gd,&gm,"C:\\TC\\BGI");
8.
9.     int toggle=0;
10.    while(1){
11.        for(int y=30;y<=400;y+=2)
12.        {
13.            pieslice(300,y,0,360,30);
14.            line(0,430,1000,430);
15.            delay(3);
16.            cleardevice();
17.        }
18.        for(int f=400;f>=30;f-=2)
19.        {
20.            pieslice(300,f,0,360,30);
21.            line(0,430,1000,430);
22.            delay(3);
23.            cleardevice();
24.            //    delay(30);
25.        }
26.    }
27.    outtext("press any key to exit");
28.    getch();
29.    closegraph();
30.
31.    return 0;
32.
33. }
```


