Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

Experiment no 1

```
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>

classpixel {
    public:
        floatx, y, length, dx, dy;
        intp;
};

classpixel1 :publicpixel {
    public:
        voidDDA(float, float, float, float);
        voidbresen(float, float, float, float);
        intsign(float);
};

intpixel1::sign(floatx) {if
    (x<0)
        return -1;
    elseif (x == 0)
        return0;
    else
        return1;
}

voidpixel1::DDA(floatx1, floaty1, floatx2, floaty2) {dx =
    abs(x2 - x1);
    dy = abs(y2 - y1);

    if (dx>dy)
        length = dx;
    else
        length = dy;
    dx = (x2 - x1) / length;dy
    = (y2 - y1) / length;

    x = x1 + 0.5 * sign(dx);y
    = y1 + 0.5 * sign(dy);


    line(0, 240, 640, 240);  // X-axis
    line(320, 0, 320, 480);  // Y-axis

    for (inti = 0; i<length; i++) {x =
        x + dx;
        y = y + dy; putpixel(x,
        y, WHITE);
    }
}

voidpixel1::bresen(floatx1, floaty1, floatx2, floaty2) {inttemp,
    exchange_flag = 0;

    dx = abs(x2 - x1);dy
    = abs(y2 - y1);

    ints1 = sign(x2 - x1);
    ints2 = sign(y2 - y1);

    x = x1;y
    = y1;

    if (dy>dx) {
        temp = dx;
        dx = dy; dy
        = temp;
        exchange_flag = 1;
    } else {
        exchange_flag = 0;
    }

    p = 2 * dy - dx;

    line(0, 240, 640, 240);  // X-axis
    line(320, 0, 320, 480);  // Y-axis

    inti = 0;
    while (i<=  dx)  { putpixel(abs(x),
        abs(y), 15);


        if (p>= 0) {
            if (exchange_flag == 1)
                x = x + s1;
            else
                y = y + s2;p
            = p - 2 * dx;
        }


        if (exchange_flag == 1)y
            = y + s2;
        else
            x = x + s1;
```

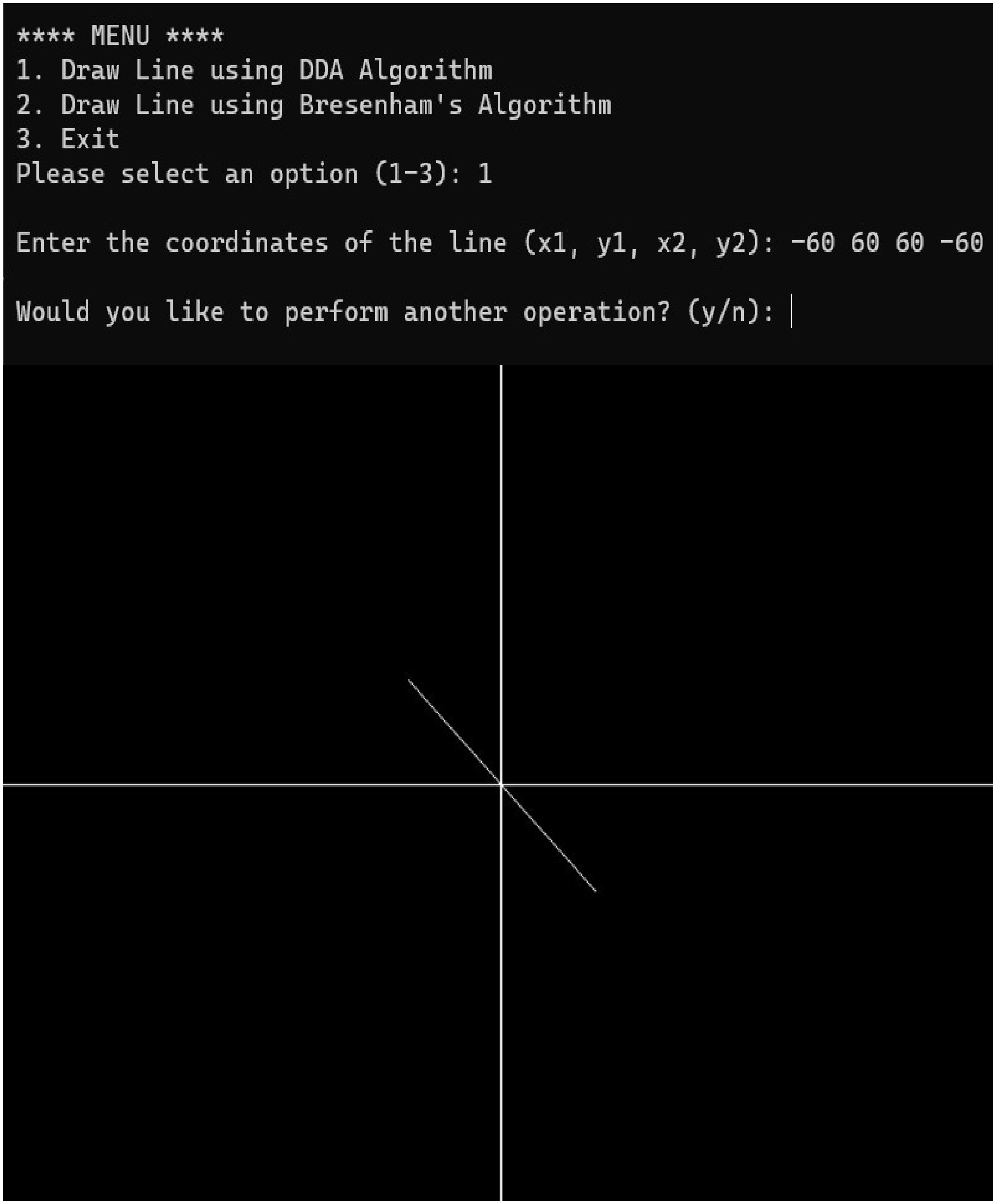
```
          p = p + 2 * dy;
          i++;
      }
 }


 intmain() {
      intgd = DETECT, gm;
      initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

      pixel1s;
      floatx1, y1, x2, y2;
      charans;
      intch;


 do {
      cout<<"\n**** MENU ****";
      cout<<"\n1. Draw Line using DDA Algorithm"; cout<<"\n2.
      Draw Line using Bresenham's Algorithm";cout<<"\n3.
      Exit";
      cout<<"\nPlease select an option (1-3): ";
      cin>>ch;


      switch  (ch)  {
          case1:
               cout<<"\nEnter  the  coordinates  of  the  line  (x1,  y1,  x2,  y2):  ";
               cin>>x1>>y1>>x2>>y2;
               x1 += 320; y1 = 240 - y1;x2
               +=  320;  y2  =  240  -  y2;
               s.DDA(x1, y1, x2, y2);
               break;


          case2:
               cout<<"\nEnter the coordinates of the line (x1, y1, x2, y2): ";
               cin>>x1>>y1>>x2>>y2;
               x1 += 320; y1 = 240 - y1;
               x2 += 320; y2 = 240 - y2;
               s.bresen(x1, y1, x2, y2);
               break;

          case3:
               cout<<"\nExiting the program.";
               break;

          default:
               cout<<"\nInvalid option. Please select a valid option (1-3).";
      }
```

```
    if (ch != 3) {
        cout<<"\nWould you like to perform another operation? (y/n): ";
        cin>>ans;
    }
}

while (ans == 'y' || ans == 'Y');

    getch();
    closegraph();
    return0;
}
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
**** MENU ****
1. Draw Line using DDA Algorithm
2. Draw Line using Bresenham's Algorithm
3. Exit
Please select an option (1-3): 1

Enter the coordinates of the line (x1, y1, x2, y2): -60 60 60 -60

Would you like to perform another operation? (y/n): 
```

Name: Bhupen Jitendra Chirmade

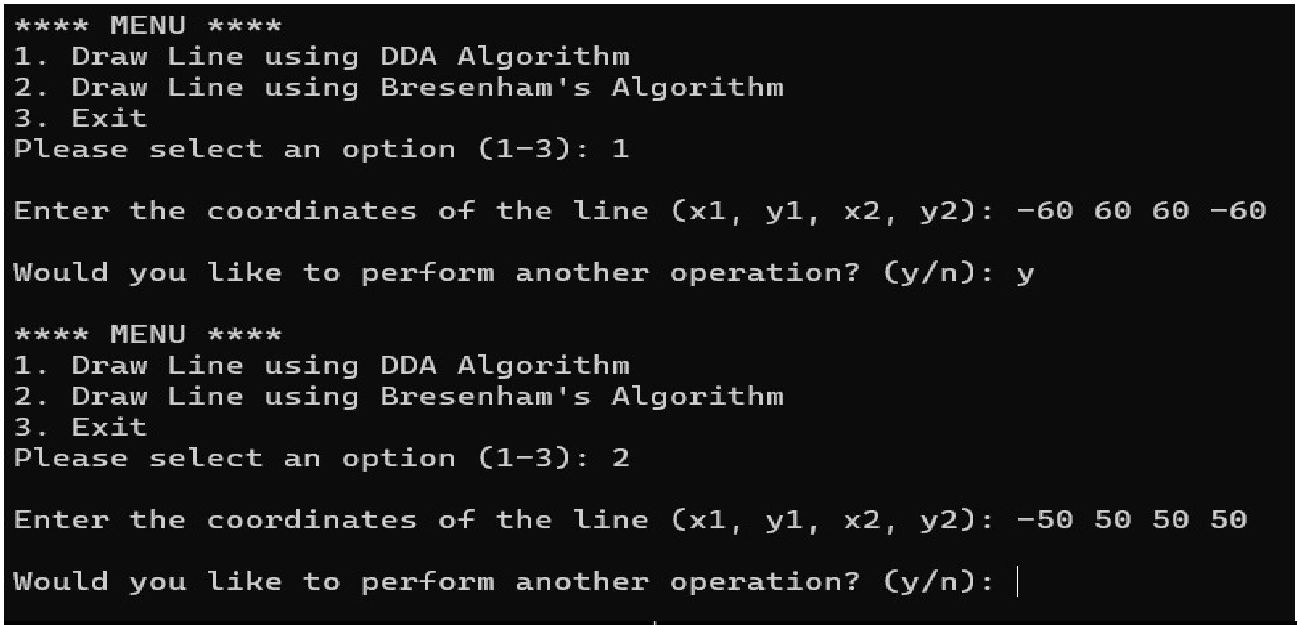
Roll Number: 22

Class: SE Computer (A)

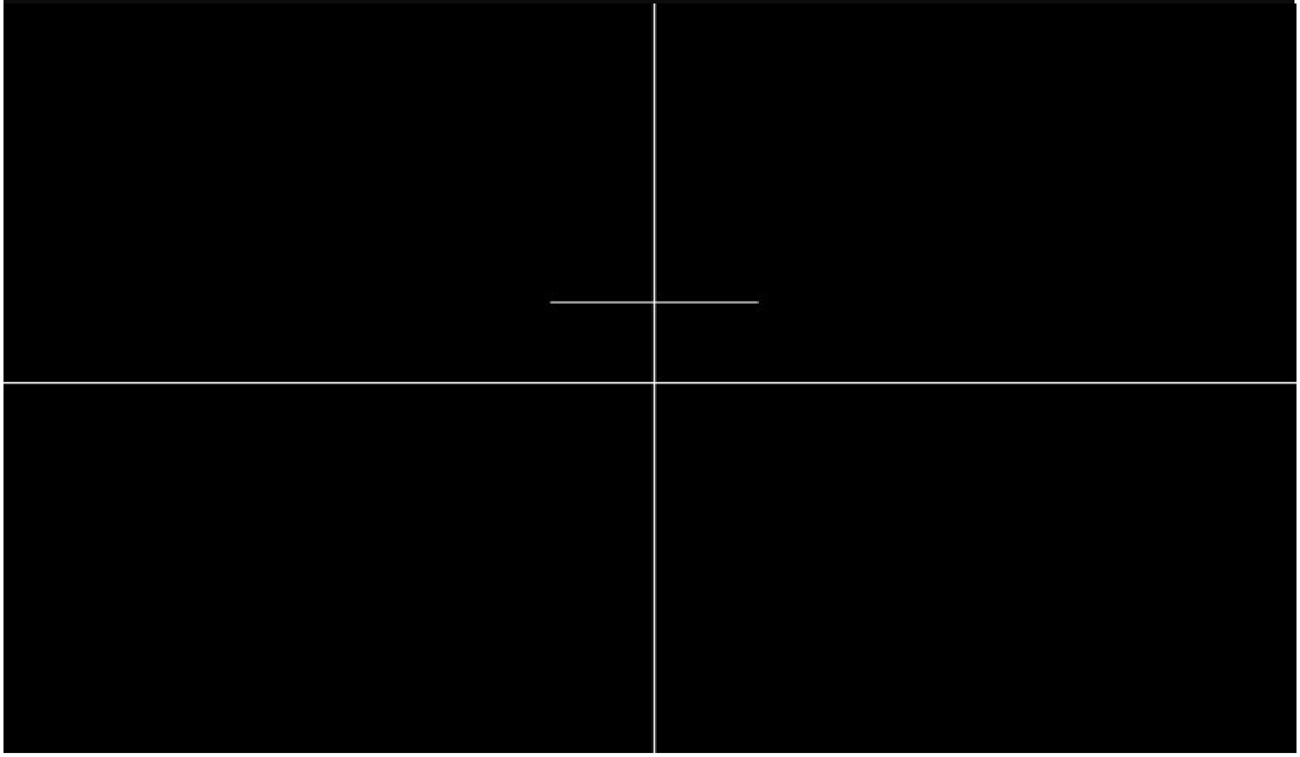
Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



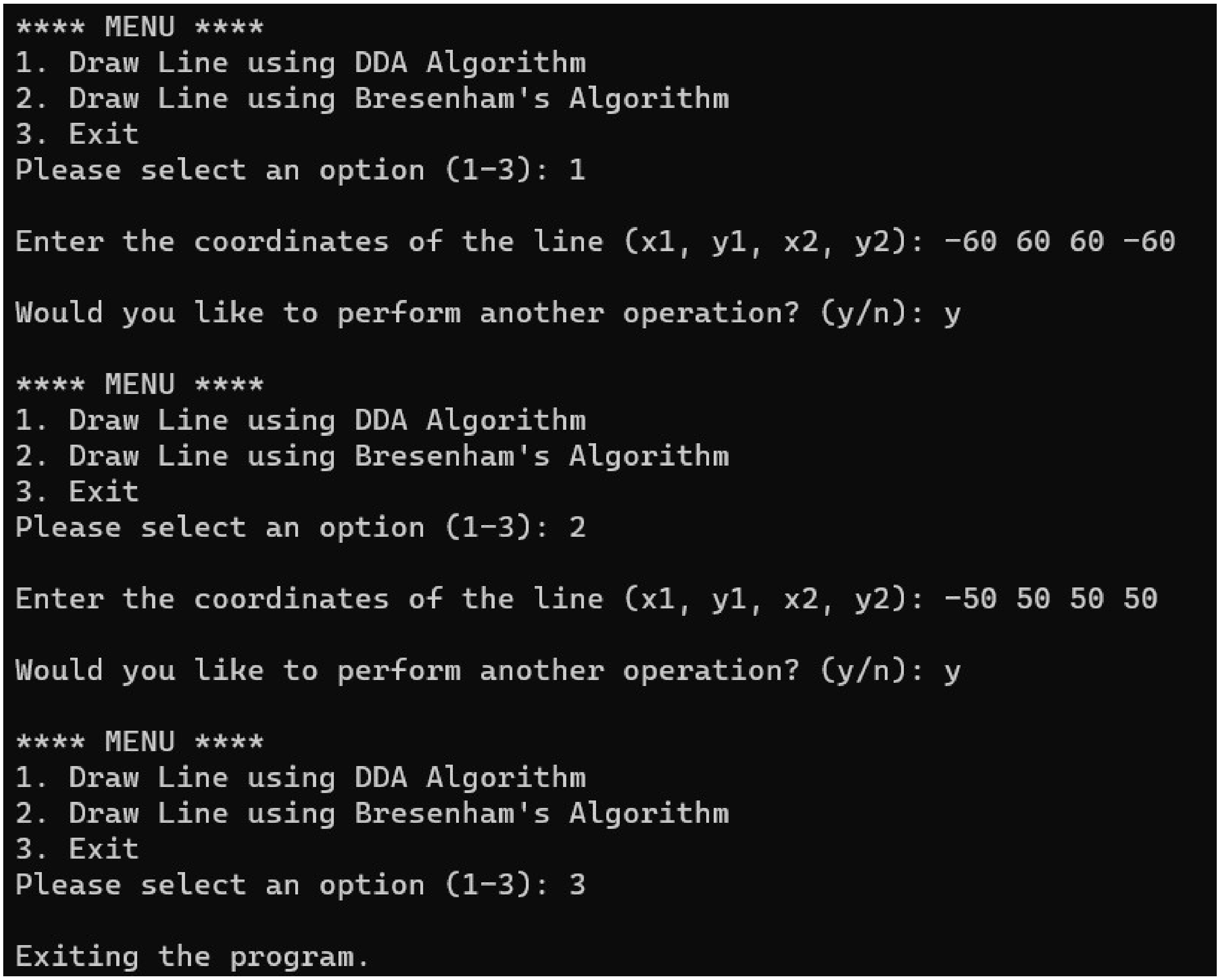
```
**** MENU ****
1. Draw Line using DDA Algorithm
2. Draw Line using Bresenham's Algorithm
3. Exit
Please select an option (1-3): 1

Enter the coordinates of the line (x1, y1, x2, y2): -60 60 60 -60

Would you like to perform another operation? (y/n): y

**** MENU ****
1. Draw Line using DDA Algorithm
2. Draw Line using Bresenham's Algorithm
3. Exit
Please select an option (1-3): 2

Enter the coordinates of the line (x1, y1, x2, y2): -50 50 50 50

Would you like to perform another operation? (y/n): 
```



Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
**** MENU ****
1. Draw Line using DDA Algorithm
2. Draw Line using Bresenham's Algorithm
3. Exit
Please select an option (1-3): 1

Enter the coordinates of the line (x1, y1, x2, y2): -60 60 60 -60

Would you like to perform another operation? (y/n): y

**** MENU ****
1. Draw Line using DDA Algorithm
2. Draw Line using Bresenham's Algorithm
3. Exit
Please select an option (1-3): 2

Enter the coordinates of the line (x1, y1, x2, y2): -50 50 50 50

Would you like to perform another operation? (y/n): y

**** MENU ****
1. Draw Line using DDA Algorithm
2. Draw Line using Bresenham's Algorithm
3. Exit
Please select an option (1-3): 3

Exiting the program.
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

Experiment no 2

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>

classA {
     public:
          intx, y, x1, y1, r, d;
};

classB :publicA {
     public:
          voidgetdata();
          voiddraw();
};

voidB::getdata() {
     cout<<"\nEnter center coordinates (x, y): ";cin>>x1>>y1;
     x1 = x1 + 320;y1 =
     240 - y1;

     cout<<"\nEnter radius of circle: ";cin>>r;
}

voidB::draw() {
     d = 3 - 2 * r;  // d is the decision parameterx = 0;
     y = r;

     line(320, 0, 320, 480);  // Y-axis
     line(0, 240, 640, 240);  // X-axis

     do {
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

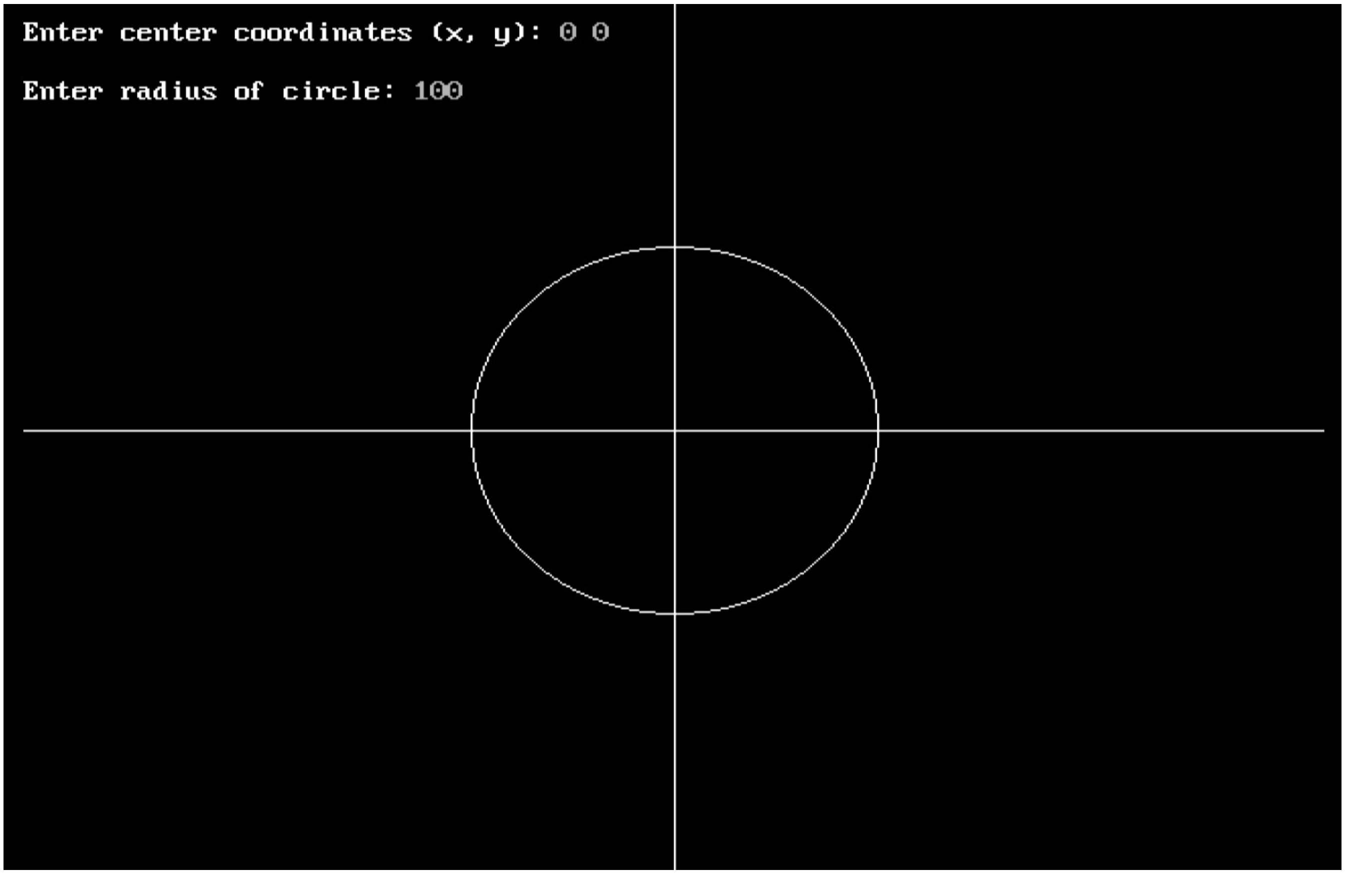
Class: SE Computer (A)

```
putpixel(x1 + x, y1 + y, 15);putpixel(x1 + x, y1
- y, 15);putpixel(x1 + y, y1 + x, 15);putpixel(x1
+ y, y1 - x, 15);putpixel(x1 - x, y1 + y, 15);
putpixel(x1 - x, y1 - y, 15);putpixel(x1 - y, y1
+ x, 15);putpixel(x1 - y, y1 - x, 15);

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

 intmain() {
      intgd = DETECT, gm;
      initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

      Bobj1; obj1.getdata();
      obj1.draw();

      getch(); closegraph();
      return0;
 }
```

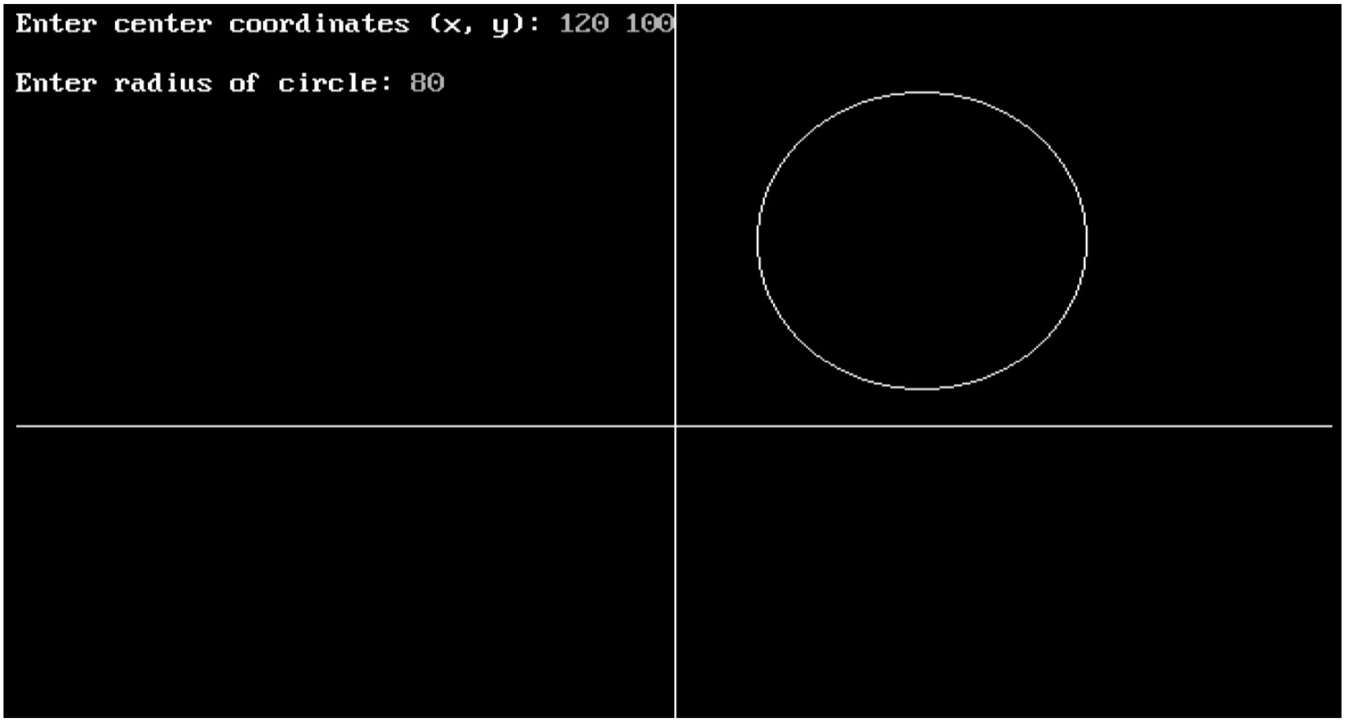
Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
Enter center coordinates (x, y): 0 0

Enter radius of circle: 100
```



```
Enter center coordinates (x, y): 120 100

Enter radius of circle: 80
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

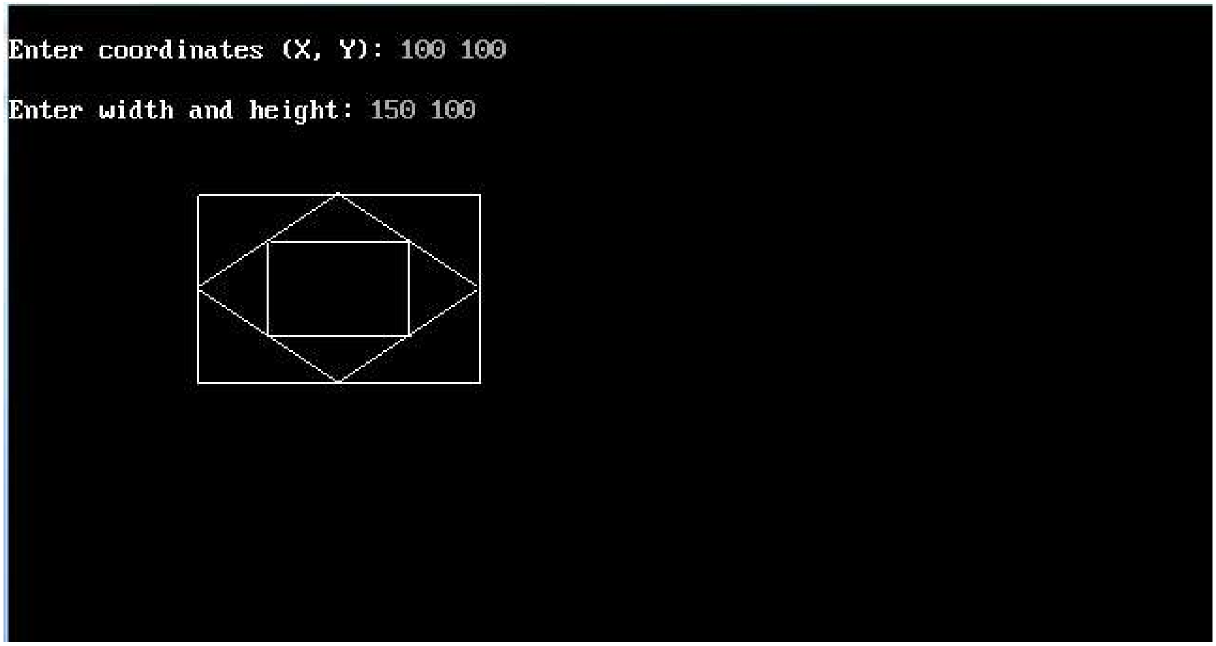
Experiment no 3

```
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>

classpixel1 {
    public:
        float x, y, length, dx, dy, height, width, X, Y;
        voidDDA(floatx1, floaty1, floatx2, floaty2);
        voidpattern();
        intsign(floatvalue);
};

intpixel1::sign(floatvalue) {
    if (value<0)
        return -1;
    elseif (value == 0)
        return0;
    else
        return1;
}

voidpixel1::DDA(floatx1, floaty1, floatx2, floaty2) {dx =
    abs(x2 - x1);
    dy = abs(y2 - y1);

    length = (dx >dy) ? dx : dy;

    dx = (x2 - x1) / length;
    dy = (y2 - y1) / length;

    x = x1 + 0.5 * sign(dx);y
    = y1 + 0.5 * sign(dy);
    for (inti = 0; i< length; i++) {
        delay(1);
        x += dx;y +=
        dy;

        putpixel(x, y, WHITE);
    }
}

voidpixel1::pattern() {
    cout<<"\nEnter coordinates (X, Y): ";cin>>
    X >> Y;

    cout<<"\nEnter width and height: ";cin>>
    width >> height;

    // Outer Rectangle
    DDA(X, Y, X + width, Y);                 // Top edge
    DDA(X, Y, X, Y + height);                // Left edge DDA(X, Y
    + height, X + width, Y + height); // Bottom edgeDDA(X + width,
    Y, X + width, Y + height);  // Right edge

    // Diagonal Cross Lines
    DDA(X, Y + height / 2, X + width / 2, Y);        // Top-left  to  center DDA(X, Y
    + height / 2, X + width / 2, Y + height); // Bottom-left to centerDDA(X + width, Y
    + height / 2, X + width / 2, Y);  // Top-right to center
    DDA(X + width, Y + height / 2, X + width / 2, Y + height); // Bottom-right to ctr

    // Inner Rectangle
    DDA(X + width / 4, Y + height / 4, X + 3 * width / 4, Y + height / 4);
    DDA(X + width / 4, Y + 3 * height / 4, X + 3 * width / 4, Y + 3 * height / 4);DDA(X +
    width / 4, Y + height / 4, X + width / 4, Y + 3 * height / 4);
    DDA(X + 3 * width / 4, Y + height / 4, X + 3 * width / 4, Y + 3 * height / 4);
}

intmain() {
    intgd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

    pixel1 s;
    s.pattern();
    delay(10000);

    getch();
    closegraph();
    return0;
}
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
Enter coordinates (X, Y): 100 100

Enter width and height: 150 100
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

Experiment no 4

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

voidddaline(intx1, inty1, intx2, inty2) { ints, m, dx, dy;
     floatxi, yi, x, y;

     dx = x2 - x1; dy = y2
     - y1;

     if (abs(dx) >abs(dy)) s =
          abs(dx);
     else
          s = abs(dy);

     xi = dx / (float)s; yi = dy /
     (float)s;

     x = x1; y =
     y1;

     putpixel(x1 + 0.5, y1 + 0.5, 15); for (m = 0;


     m<s; m++) {
          x += xi;
          y += yi;
          putpixel(x + 0.5, y + 0.5, 15);

     }
}

voidfill(intx, inty) { inti;
     for (i = x; i< (x + 50); i++) { ddaline(i, y, i, y +
          50);
     }
}

intmain() {
     inti, j, c = 0;
     intgd = DETECT, gm = DETECT;

     initgraph(&gd, &gm, "C:\\Turboc3\\BGI"); cleardevice();
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

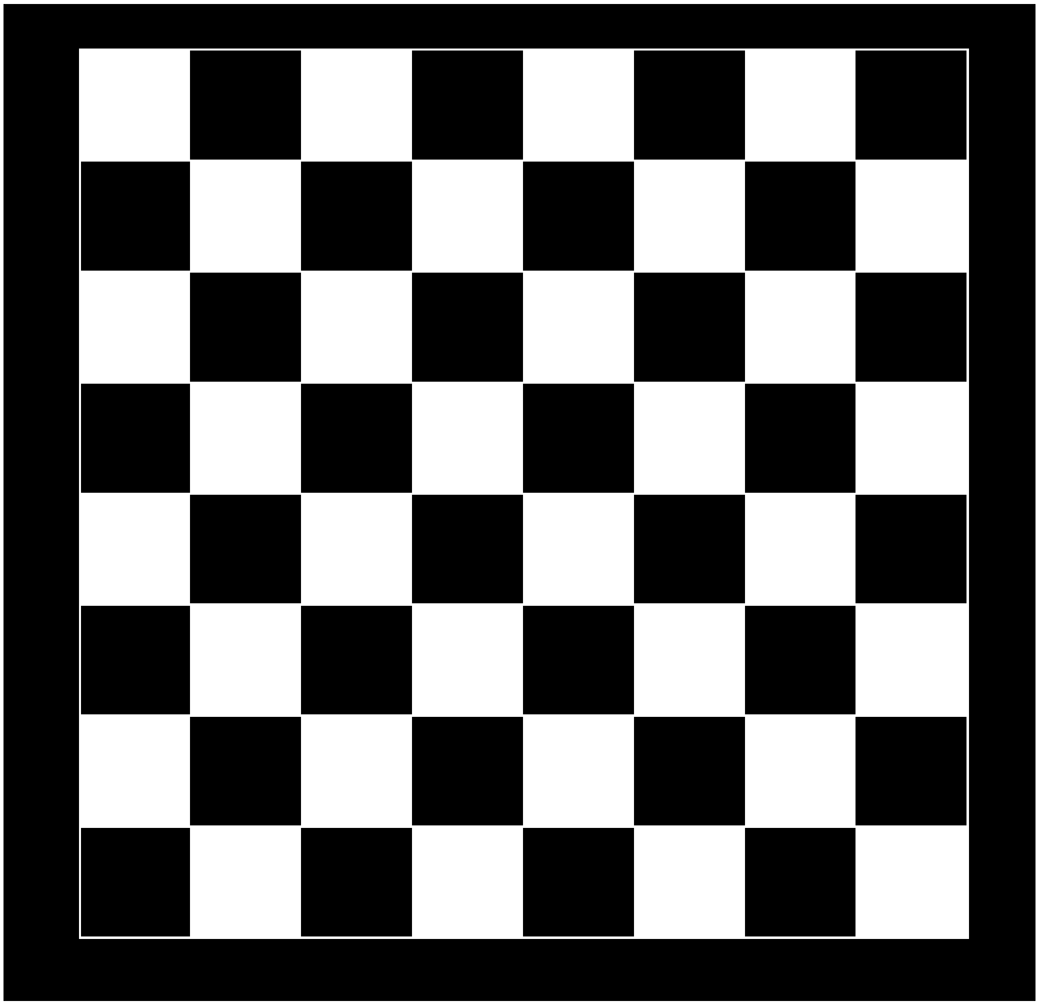
Class: SE Computer (A)

```
      // Draw the outer square ddaline(100, 50, 100,
      450);
      ddaline(100, 50, 500, 50);
      ddaline(500, 50, 500, 450);
      ddaline(100, 450, 500, 450);

      // Fill the pattern
      for (i = 100; i<500; i += 50) { for (j = 50;
           j<450; j += 50) {
                if (c % 2 == 0) fill(i, j);
                c++;
           } c++;
      }

      getch();
      return0;
 }
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

Experiment no 5

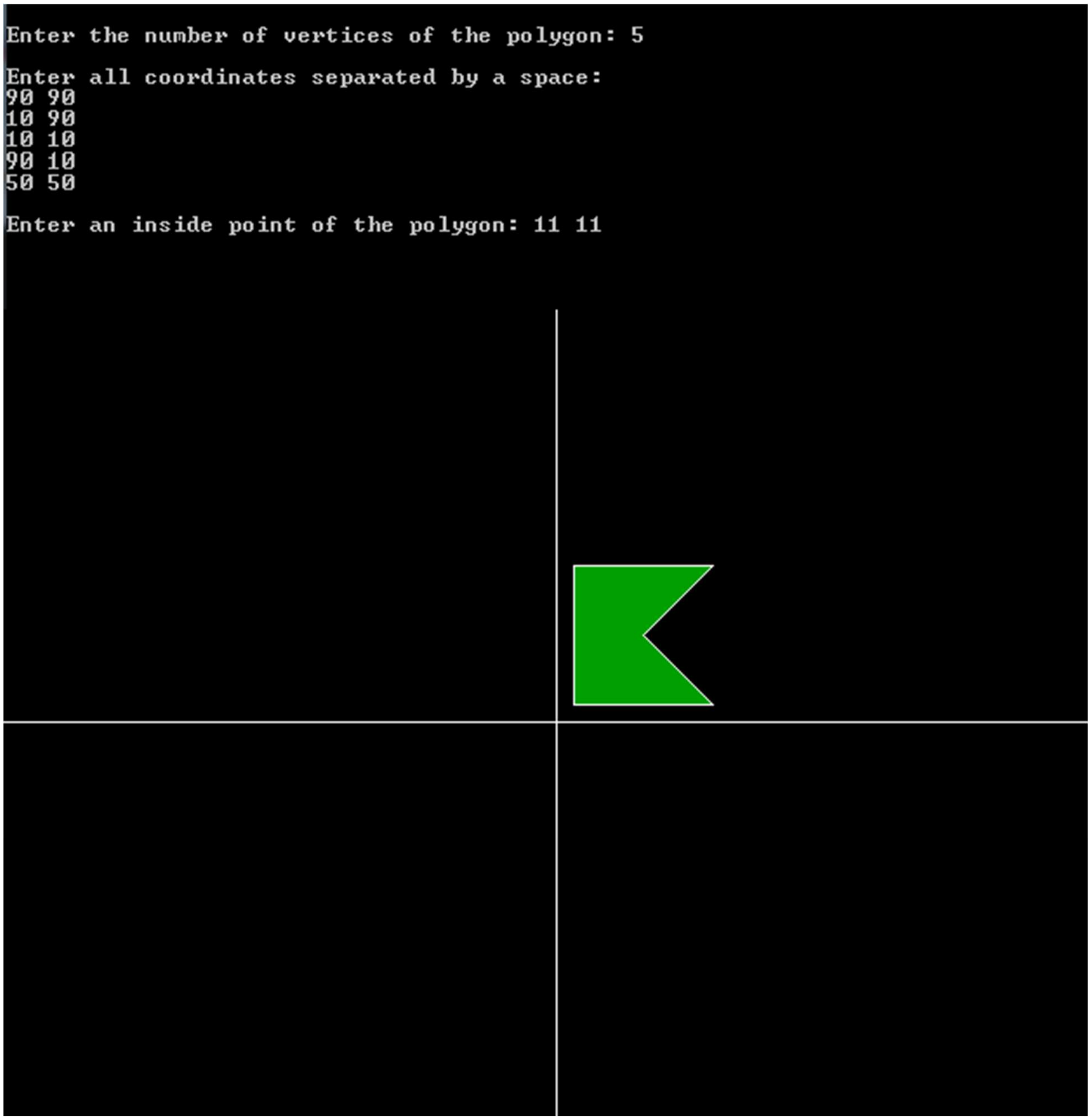
```
#include<iostream.h>

#include<graphics.h>

#include<dos.h>

#include<conio.h>

#include<math.h>
 class SeedFill { int x, y;

 public:
      void seedFill(int, int, int, int); void
      drawPolygon();
 };

 void SeedFill::drawPolygon() { int
      p[20][2], i, n;

      line(320, 0, 320, 480);
      line(0, 240, 640, 240);

      cout << "\nEnter the number of vertices of the polygon: "; cin >> n;

      cout << "\nEnter all coordinates separated by a space:\n"; for (i = 0; i <
      n; i++) {
      cin >> p[i][0] >> p[i][1];
      p[i][0] = 320 + p[i][0];
      p[i][1] = 240 - p[i][1];
      }

      p[i][0] = p[0][0];
      p[i][1] = p[0][1];
      for (i = 0; i < n; i++) {
      line(p[i][0], p[i][1], p[i + 1][0], p[i + 1][1]);
      }

      line(p[i][0], p[i][1], p[0][0], p[0][1]);
 }

 void SeedFill::seedFill(int x, int y, int oldColor, int newColor) { int color =
      getpixel(x, y);

      if (color == oldColor && color != newColor) { putpixel(x, y,
      newColor);
      delay(2);
```

```
    seedFill(x + 1, y, oldColor, newColor); seedFill(x -
    1, y, oldColor, newColor); seedFill(x, y + 1,
    oldColor, newColor); seedFill(x, y - 1, oldColor,
    newColor);
    }
}

int main() {
    int gd = DETECT, gm = DETECT; initgraph(&gd, &gm,
    "C:\\Turboc3\\BGI");

    SeedFill seedFillObject;
    seedFillObject.drawPolygon();

    int x, y;
    cout << "\nEnter an inside point of the polygon: "; cin >> x >> y;
    x = x + 320; y =
    240 - y;

    int oldColor = getpixel(x, y); int newColor
    = 2;

    seedFillObject.seedFill(x, y, oldColor, newColor); delay(1000000);
    closegraph();

    return 0;
}
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
Enter the number of vertices of the polygon: 5

Enter all coordinates separated by a space:
90 90
10 90
10 10
90 10
50 50

Enter an inside point of the polygon: 11 11
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

Experiment no 6

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

static int LEFT = 1, RIGHT = 2, BOTTOM = 4, TOP = 8;
int xmin, ymin, xmax, ymax;

int getCode(int x, int y) {
    int code = 0;
    if (y > ymax) code |= TOP;
    if (y < ymin) code |= BOTTOM;
    if (x < xmin) code |= LEFT;
    if (x > xmax) code |= RIGHT;
    return code;
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

    cout << "Enter the window's maximum and minimum coordinates (xmin, ymin, xmax, ymax): ";
    cin >> xmin >> ymin >> xmax >> ymax;
    rectangle(xmin, ymin, xmax, ymax);

    int x1, y1, x2, y2;
    cout << "Enter the line coordinates (x1, y1, x2, y2): ";
    cin >> x1 >> y1 >> x2 >> y2;
    line(x1, y1, x2, y2);
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

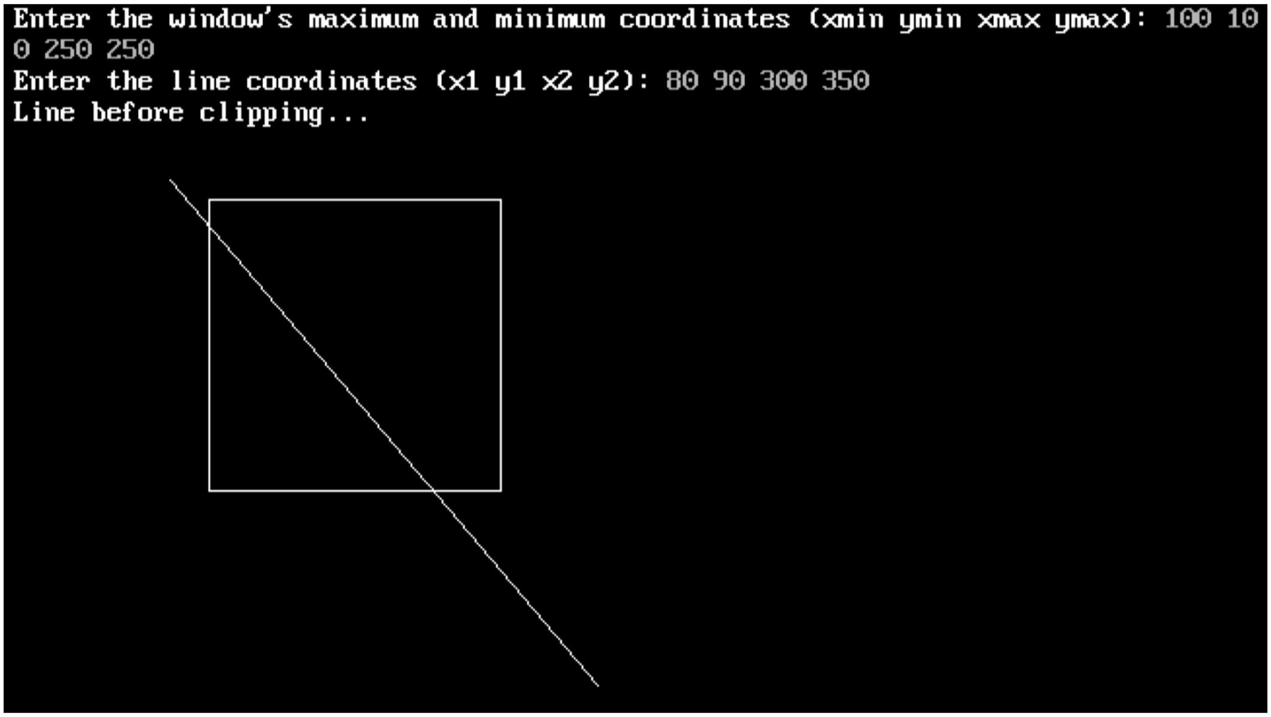
Class: SE Computer (A)

```
    int outcode1 = getCode(x1, y1);
    int outcode2 = getCode(x2, y2);
    int accept = 0;

    while (true) {
        if (outcode1 == 0 && outcode2 == 0) {
            accept = 1;
            break;
        } else if ((outcode1 & outcode2) != 0) {
            break;
        } else {
            int x, y;
            int temp;

            if (outcode1 != 0) temp = outcode1;
            else temp = outcode2;

            if (temp & TOP) {
                x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
                y = ymax;
            } else if (temp & BOTTOM) {
                x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1);
                y = ymin;
            } else if (temp & RIGHT) {
                y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1);
                x = xmax;
            } else if (temp & LEFT) {
                y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1);
                x = xmin;
            }
            if (temp == outcode1) {
                x1 = x;
                y1 = y;
                outcode1 = getCode(x1, y1);
            } else {
                x2 = x;
                y2 = y;
                outcode2 = getCode(x2, y2);
            }
        }
    }

    cout << "After clipping:\n";
    cleardevice();
    rectangle(xmin, ymin, xmax, ymax);
    if (accept) {
        line(x1, y1, x2, y2);
    } else {
        cout << "Line is outside the clipping region.\n";
    }

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

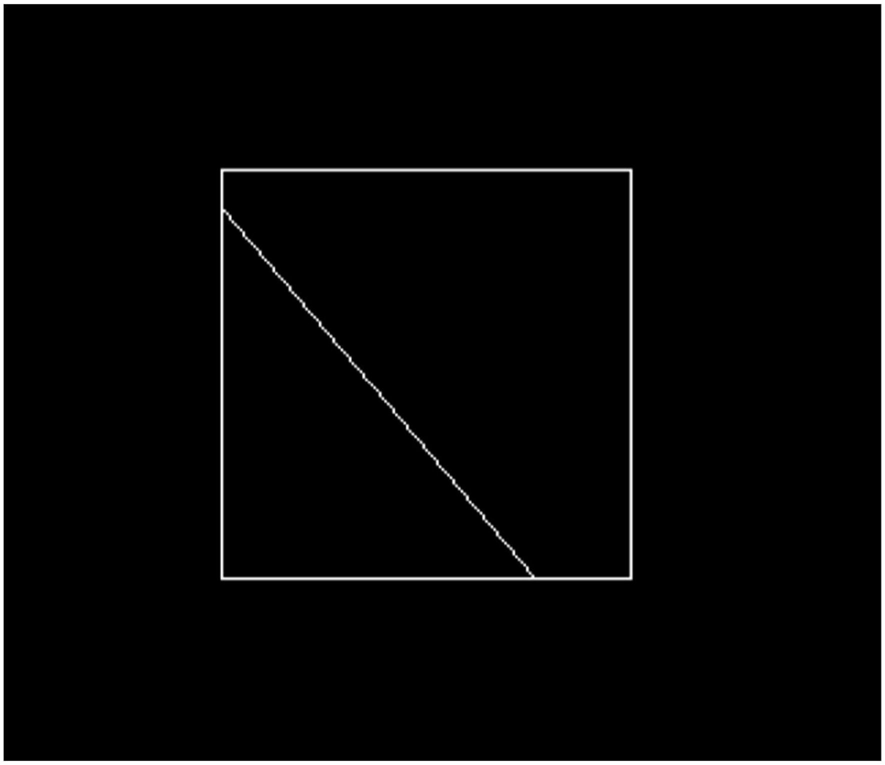
Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
Enter the window's maximum and minimum coordinates (xmin ymin xmax ymax): 100 10
0 250 250
Enter the line coordinates (x1 y1 x2 y2): 80 90 300 350
Line before clipping...
```



Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

Experiment no 8

```
#include <graphics.h>
#include <iostream>
#include <conio.h>
#include <dos.h>

using namespace std;

float points[4][2];  // Control points for the Bézier curve

// Function to draw a line and update the last point void line1(float x2,
float y2) {     line(points[0][0], points[0][1], x2, y2);     points[0][0] =
x2;     points[0][1] = y2;
}

// Function to compute Bézier curve using recursive method void bezier(float xa, float ya, float xb, float yb, float
xc, float yc, float xd, float yd, int n) {     float xab, yab, xbc, ybc, xcd, ycd, xabc, yabc, xbcd, ybcd, xabcd, yabcd;

    if (n == 0) {         line1(xb, yb);
line1(xc, yc);         line1(xd, yd);
delay(100);
    } else {
        // Calculate midpoints         xab = (xa + xb) / 2; yab = (ya + yb) / 2;
xbc = (xb + xc) / 2; ybc = (yb + yc) / 2;         xcd = (xc + xd) / 2; ycd = (yc + yd)
/ 2;         xabc = (xab + xbc) / 2; yabc = (yab + ybc) / 2;         xbcd = (xbc + xcd)
/ 2; ybcd = (ybc + ycd) / 2;         xabcd = (xabc + xbcd) / 2; yabcd = (yabc +
ybcd) / 2;

        // Recursive calls         bezier(xa, ya, xab, yab, xabc, yabc, xabcd, yabcd, n - 1);
bezier(xabcd, yabcd, xbcd, ybcd, xcd, ycd, xd, yd, n - 1);
    }
}
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

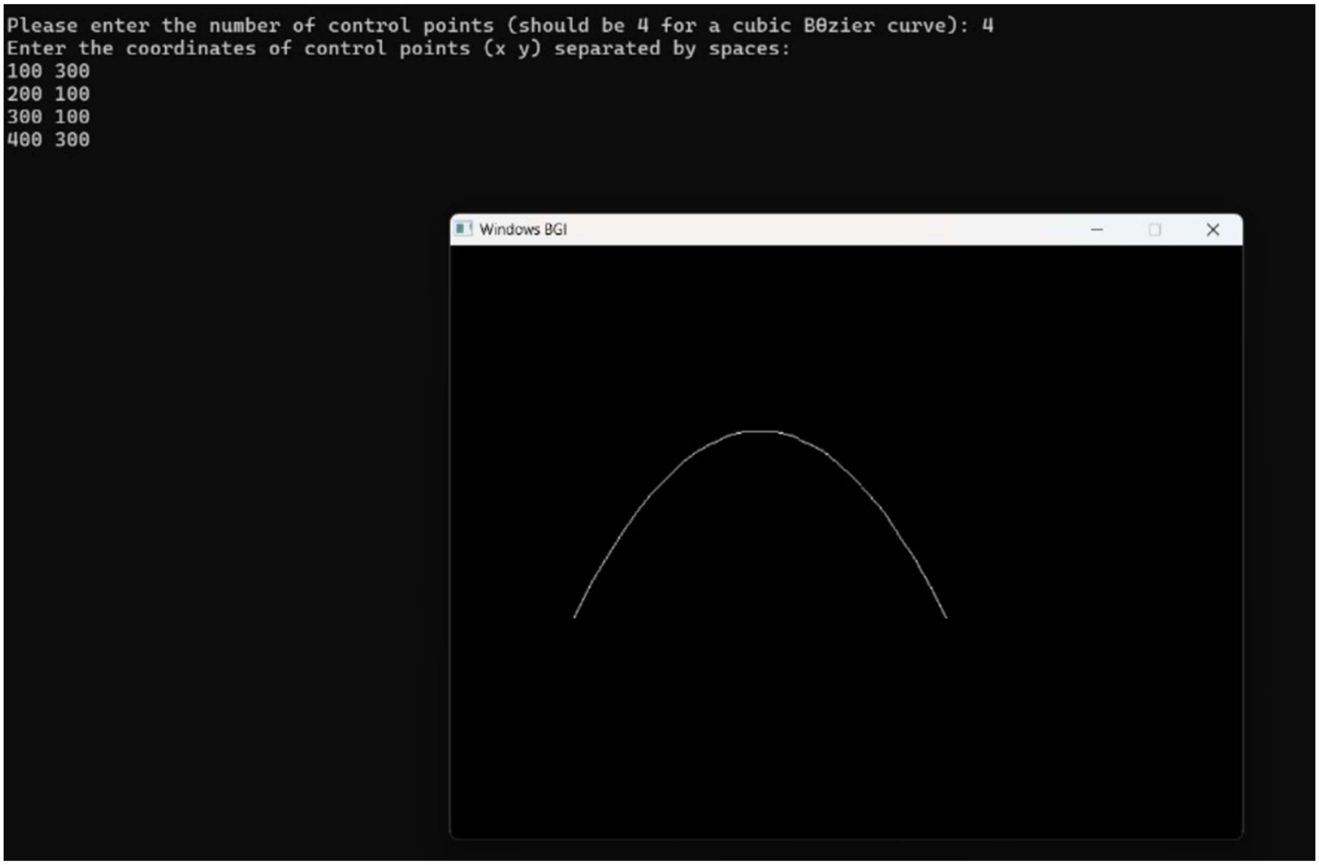
Class: SE Computer (A)

```
int main() {     int gd = DETECT, gm, n;

    // Initialize graphics     initgraph(&gd, &gm,
"c:\\turboc3\\bgi");     cleardevice();

    cout << "Please enter the number of control points (should be 4 for a cubic Bézier curve): ";     cin >> n;

    // Input control points
    cout << "Enter the coordinates of control points (x y) separated by spaces:\n";     for (int i = 0; i < n;
i++) {         cin >> points[i][0] >> points[i][1];
    }

    // Draw the Bézier curve     bezier(points[0][0], points[0][1], points[1][0], points[1][1],
points[2][0], points[2][1], points[3][0], points[3][1], n - 1);

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

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
Please enter the number of control points (should be 4 for a cubic B0zier curve): 4
Enter the coordinates of control points (x y) separated by spaces:
100 300
200 100
300 100
400 300
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

Experiment no 9

```
#include <graphics.h>
#include <iostream>
#include <cmath>
#include <conio.h>
using namespace std;
void koch(int x1, int y1, int x2, int y2, int it) {
    // Calculate the angle for the Koch curve
    float angle = 60 * M_PI / 180;  // Use M_PI from cmath for better precision
    int x3 = (2 * x1 + x2) / 3;
    int y3 = (2 * y1 + y2) / 3;
    int x4 = (x1 + 2 * x2) / 3;
    int y4 = (y1 + 2 * y2) / 3;
    // Calculate the peak of the Koch triangle
    int x = x3 + (x4 - x3) * cos(angle) - (y4 - y3) * sin(angle);
    int y = y3 + (x4 - x3) * sin(angle) + (y4 - y3) * cos(angle);
    // Recursive case
    if (it > 0) {
        koch(x1, y1, x3, y3, it - 1);
        koch(x3, y3, x, y, it - 1);
        koch(x, y, x4, y4, it - 1);
        koch(x4, y4, x2, y2, it - 1);
    } else {
        // Draw the line segments
        line(x1, y1, x3, y3);
        line(x3, y3, x, y);
        line(x, y, x4, y4);
        line(x4, y4, x2, y2);
    }
    delay(100); // Optional delay for visualization
}
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

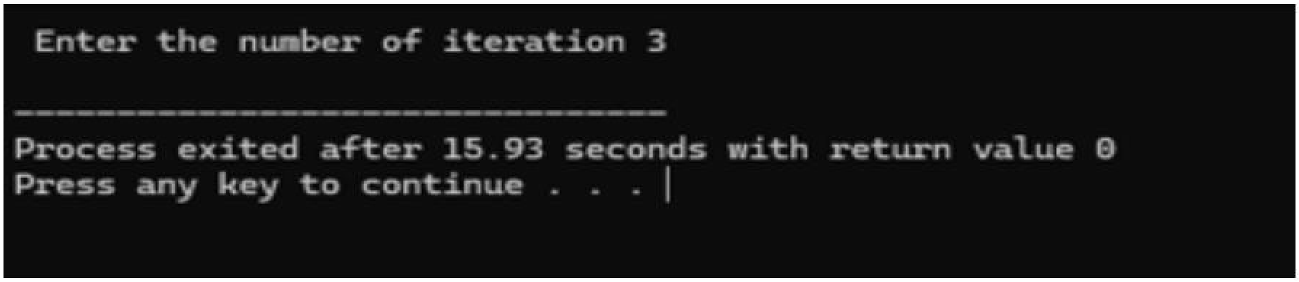
Class: SE Computer (A)

```
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    int x1 = 100, y1 = 300, x2 = 400, y2 = 300;  // Starting points for the Koch curve
    int iteration;

    cout << "\nEnter the number of iterations: ";
    cin >> iteration;

    koch(x1, y1, x2, y2, iteration);  // Generate the Koch curve

    delay(3000); // Delay before closing the window
    closegraph();
    return 0;
}
```

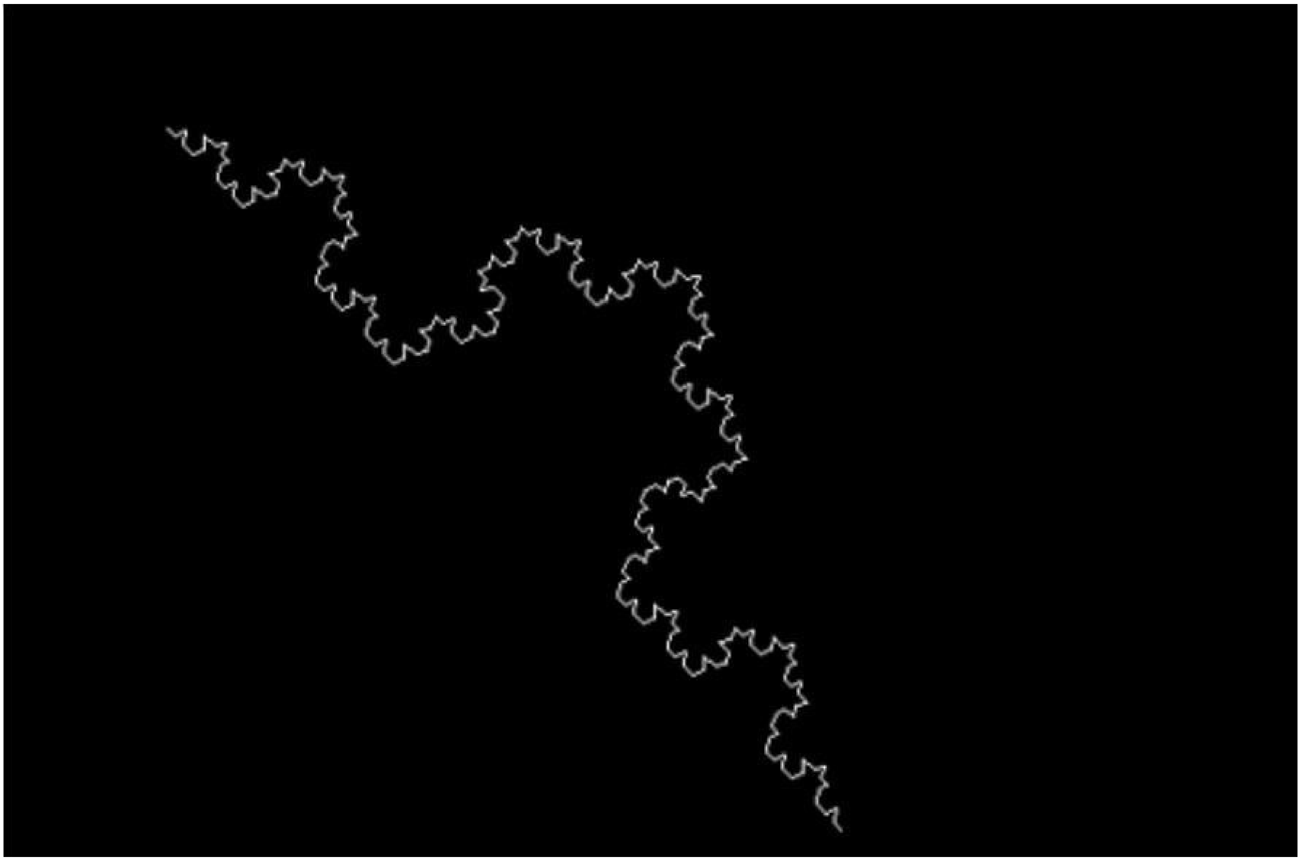
Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
 Enter the number of iteration 3

--------------------------------
Process exited after 15.93 seconds with return value 0
Press any key to continue . . .
```



Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

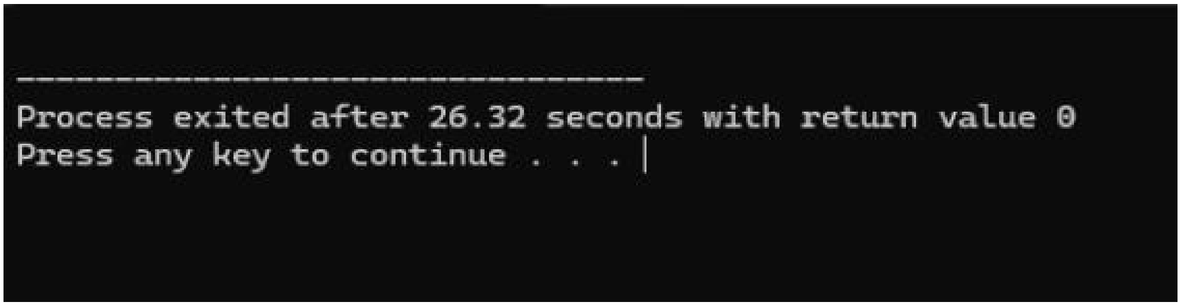
Experiment no 11

```
#include <graphics.h>
#include <conio.h>
int main() {
    // Initialize graphics mode
    int gd = DETECT, gm;
    // Initialize variables
    int i, maxx, midy;
    // Initialize graphics mode
    initgraph(&gd, &gm, NULL);
    // Get maximum pixel in horizontal axis
    maxx = getmaxx();
    // Get mid pixel in vertical axis
    midy = getmaxy() / 2;
    // Animation loop
    for (i = 0; i < maxx - 120; i += 4) {
        // Clear screen
        cleardevice();
        // Draw a white road
        setcolor(WHITE);
        line(0, midy + 37, maxx, midy + 37);
        // Draw Car
        setcolor(YELLOW);
        line(i, midy + 23, i, midy);
        line(i, midy, 40 + i, midy - 20);
        line(40 + i, midy - 20, 80 + i, midy - 20);
        line(80 + i, midy - 20, 100 + i, midy);
        line(100 + i, midy, 120 + i, midy);
        line(120 + i, midy, 120 + i, midy + 23);
        line(0 + i, midy + 23, 18 + i, midy + 23);
        arc(30 + i, midy + 23, 0, 180, 12);
        line(42 + i, midy + 23, 78 + i, midy + 23);
        arc(90 + i, midy + 23, 0, 180, 12);
        line(102 + i, midy + 23, 120 + i, midy + 23);
        line(28 + i, midy, 43 + i, midy - 15);
        line(43 + i, midy - 15, 57 + i, midy - 15);
        line(57 + i, midy - 15, 57 + i, midy);
        line(57 + i, midy, 28 + i, midy);
        line(62 + i, midy - 15, 77 + i, midy - 15);
        line(77 + i, midy - 15, 92 + i, midy);
        line(92 + i, midy, 62 + i, midy);
        line(62 + i, midy, 62 + i, midy - 15);
        // Fill the car body
        floodfill(5 + i, midy + 22, YELLOW);
        setcolor(BLUE);
        // Draw Wheels
        circle(30 + i, midy + 25, 9);
        circle(90 + i, midy + 25, 9);
        floodfill(30 + i, midy + 25, BLUE);
        floodfill(90 + i, midy + 25, BLUE);
        // Add delay of 50 milliseconds
        delay(50);
    }
    // Wait for a key press
    getch();
    // Close graphics window
    closegraph();
    return 0;
}
```

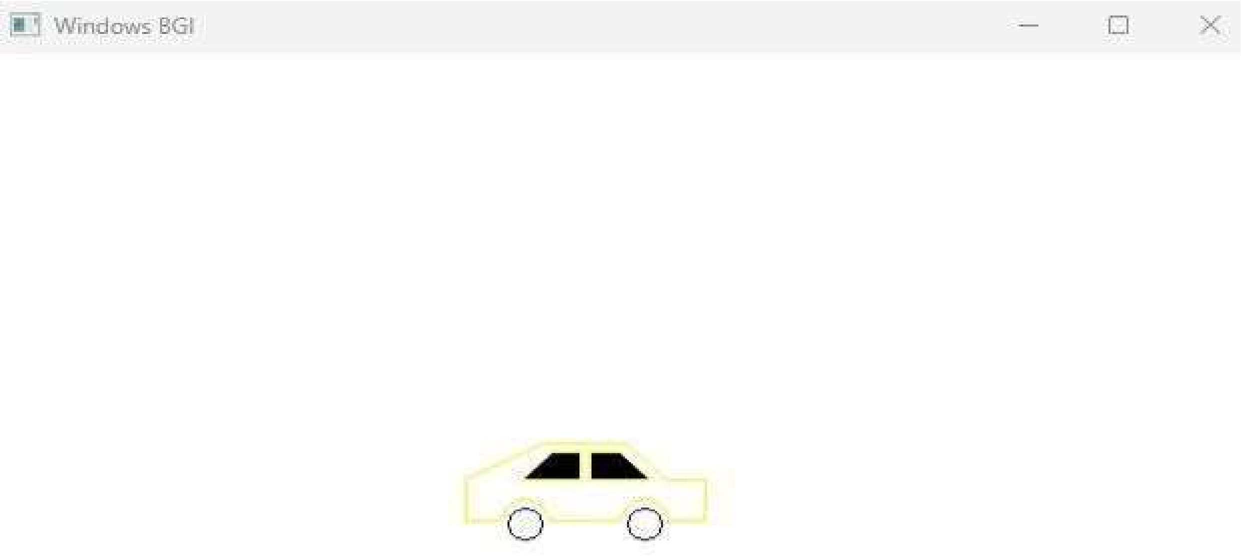
Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
--------------------------------
Process exited after 26.32 seconds with return value 0
Press any key to continue . . .
```



Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

Experiment no 7

```
#include <iostream>
#include <math.h>
#include <GL/glut.h>
using namespace std;

typedef float Matrix4[4][4];
Matrix4 theMatrix;
static GLfloat input[8][3] = {
    {40, 40, -50}, {90, 40, -50}, {90, 90, -50}, {40, 90, -50},
    {30, 30, 0},   {80, 30, 0},   {80, 80, 0},   {30, 80, 0}
};
float output[8][3];
float tx, ty, tz;
float sx, sy, sz;
float angle;
int choice, choiceRot;

void setIdentityM(Matrix4 m) {
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            m[i][j] = (i == j);
}

void translate(int tx, int ty, int tz) {
    for (int i = 0; i < 8; i++) {
        output[i][0] = input[i][0] + tx;
        output[i][1] = input[i][1] + ty;
        output[i][2] = input[i][2] + tz;
    }
}
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

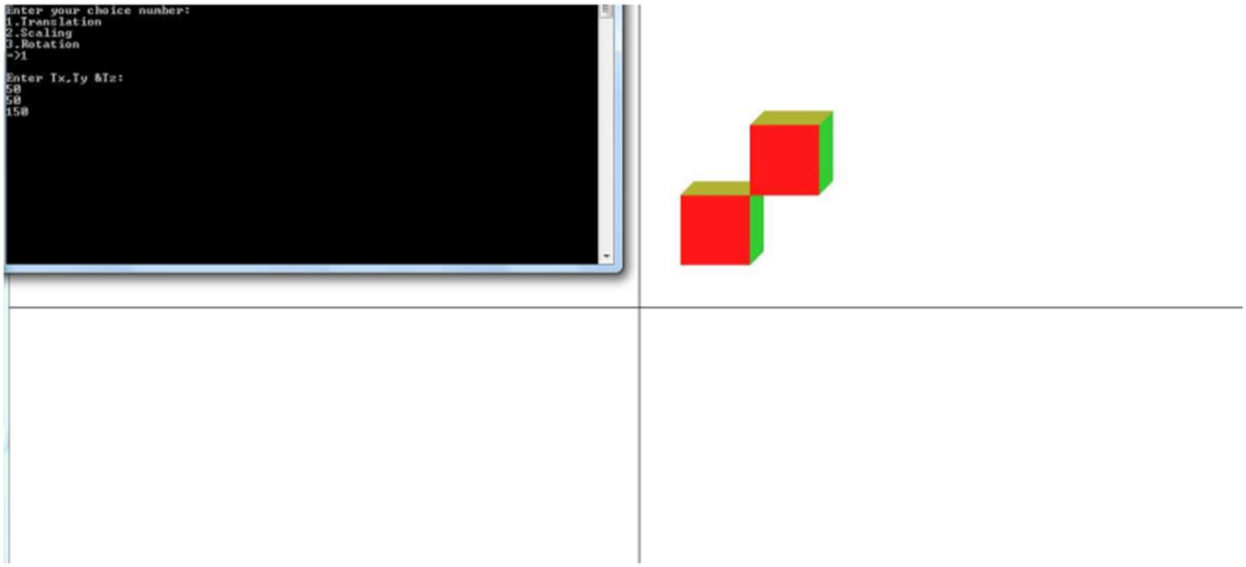
```
void scale(int sx, int sy, int sz) {
    setIdentityM(theMatrix);
    theMatrix[0][0] = sx;
    theMatrix[1][1] = sy;
    theMatrix[2][2] = sz;

    for (int i = 0; i < 8; i++) {
        output[i][0] = input[i][0] * theMatrix[0][0];
        output[i][1] = input[i][1] * theMatrix[1][1];
        output[i][2] = input[i][2] * the Matrix[2][2];
    }
}

void RotateX(float angle) {
    angle = angle * 3.142 / 180;
    setIdentityM(theMatrix);
    theMatrix[1][1] = cos(angle);
    theMatrix[2][1] = sin(angle);
    theMatrix[2][2] = cos(angle);
}

void RotateY(float angle) {
    angle = angle * 3.142 / 180;
    setIdentityM(theMatrix);
    theMatrix[0][0] = cos(angle);
    theMatrix[0][2] = -sin(angle);
    theMatrix[2][0] = sin(angle);
    theMatrix[2][2] = cos(angle);
}

void RotateZ(float angle) {
    angle = angle * 3.142 / 180;
    setIdentityM(theMatrix);
    theMatrix[0][0] = cos(angle);
    theMatrix[0][1] = sin(angle);
    theMatrix[1][0] = -sin(angle);
    theMatrix[1][1] = cos(angle);
}

void multiplyM() {
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 3; j++) {
            output[i][j] = 0;
            for (int k = 0; k < 3; k++) {
                output[i][j] = output[i][j] + input[i][k] * theMatrix[k][j];
            }
        }
    }
}

void Axes(void) {
    glColor3f(0.0, 0.0, 0.0);
    glBegin(GL_LINES);
    glVertex2s(-1000, 0);
    glVertex2s(1000, 0);
    glEnd();
    glBegin(GL_LINES);
    glVertex2s(0, -1000);
    glVertex2s(0, 1000);
    glEnd();
}

void draw(float a[8][3]) {
    glBegin(GL_QUADS);
    glColor3f(0.7, 0.4, 0.5);
    glVertex3fv(a[0]);
    glVertex3fv(a[1]);
    glVertex3fv(a[2]);
    glVertex3fv(a[3]);
    glColor3f(0.8, 0.2, 0.4);
    glVertex3fv(a[0]);
    glVertex3fv(a[1]);
    glVertex3fv(a[5]);
    glVertex3fv(a[4]);
    glColor3f(0.3, 0.6, 0.7);
    glVertex3fv(a[0]);
    glVertex3fv(a[4]);
    glVertex3fv(a[7]);
    glVertex3fv(a[3]);
    glColor3f(0.2, 0.8, 0.2);
    glVertex3fv(a[1]);
    glVertex3fv(a[2]);
    glVertex3fv(a[6]);
    glVertex3fv(a[5]);
    glColor3f(0.7, 0.7, 0.2);
    glVertex3fv(a[2]);
    glVertex3fv(a[3]);
    glVertex3fv(a[7]);
    glVertex3fv(a[6]);
    glColor3f(1.0, 0.1, 0.1);
    glVertex3fv(a[4]);
    glVertex3fv(a[5]);
    glVertex3fv(a[6]);
    glVertex3fv(a[7]);
    glEnd();
}

void init() {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glOrtho(-454.0, 454.0, -250.0, 250.0, -250.0, 250.0);
    glEnable(GL_DEPTH_TEST);
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    Axes();
    glColor3f(1.0, 0.0, 0.0);
    draw(input);
    setIdentityM(theMatrix);
    switch (choice) {
        case 1:
            translate(tx, ty, tz);
            break;
        case 2:
            scale(sx, sy, sz);
            multiplyM();
            break;
        case 3:
            switch (choiceRot) {
                case 1:
                    RotateX(angle);
                    break;
                case 2:
                    RotateY(angle);
                    break;
                case 3:
                    RotateZ(angle);
                    break;
                default:
                    break;
            }
            multiplyM();
            break;
    }
    draw(output);
    glFlush();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(1362, 750);
    glutInitWindowPosition( 0, 0);
    glutCreateWindow("3D TRANSFORMATIONS");
    init();
    cout << "Enter your choice number:\n1.Translation\n2.Scaling\n3.Rotation\n=>";
    cin >> choice;
    switch (choice) {
        case 1:
            cout << "\nEnter Tx,Ty &Tz: \n";
            cin >> tx >> ty >> tz;
            break;
        case 2:
            cout << "\nEnter Sx,Sy & Sz: \n";
            cin >> sx >> sy >> sz;
            break;
        case 3:
            cout << "Enter your choice for Rotation about axis:\n1.parallel to X-axis."
                 << "(y& z)\n2.parallel to Y-axis.(x& z)\n3.parallel to Z-axis."
                 << "(x& y)\n =>";
            cin >> choiceRot;
            switch (choiceRot) {
                case 1:
                    cout << "\nENter Rotation angle: ";
                    cin >> angle;
                    break;
                case 2:
                    cout << "\nENter Rotation angle: ";
                    cin >> angle;
                    break;
                case 3:
                    cout << "\nENter Rotation angle: ";
                    cin >> angle;
                    break;
                default:
                    break;
            }
            break;
        default:
            break;
    }
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

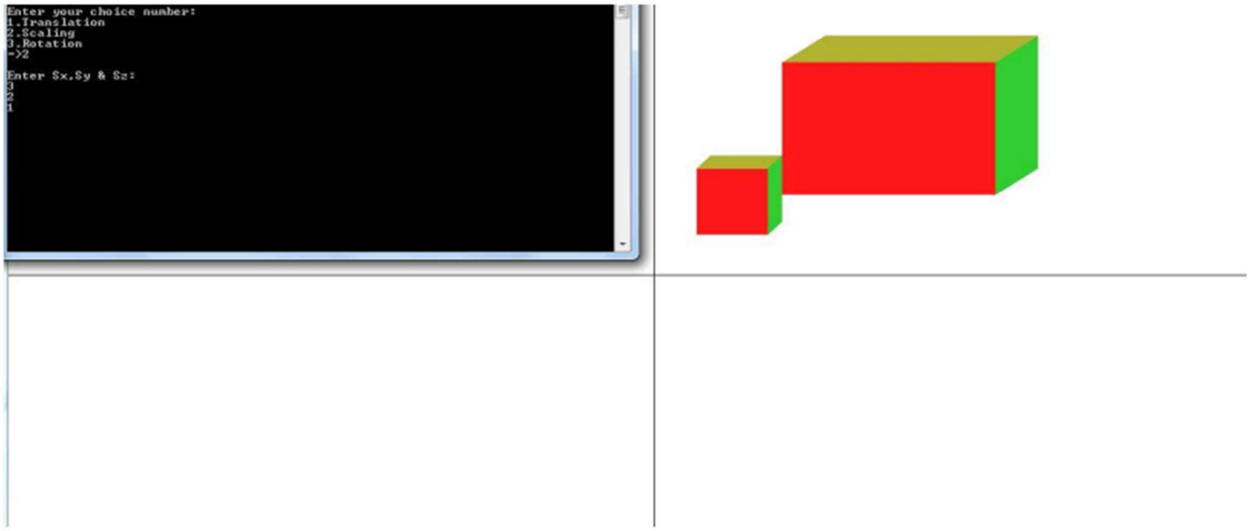
Class: SE Computer (A)

1. Translation



```
Enter your choice number:
1.Translation
2.Scaling
3.Rotation
=>1

Enter Tx,Ty &Tz:
50
50
150
```

2. Scaling



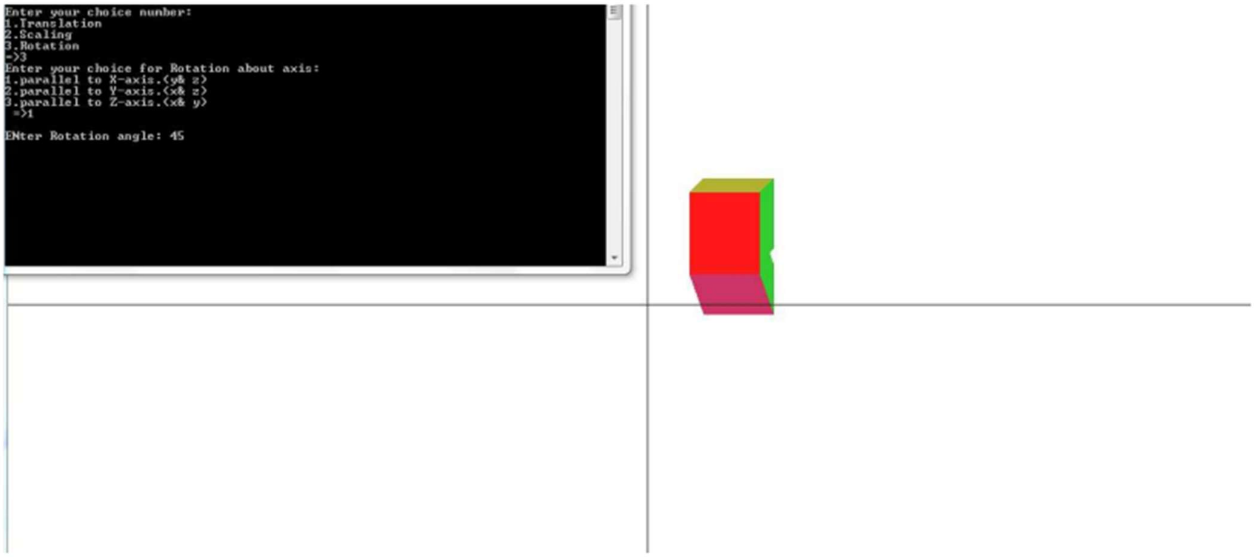
```
Enter your choice number:
1.Translation
2.Scaling
3.Rotation
->2

Enter Sx,Sy & Sz:
3
2
1
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

3. Rotation



```
Enter your choice nunber:
1.Translation
2.Scaling
3.Rotation
=>3
Enter your choice for Rotation about axis:
1.parallel to X-axis.(y& z)
2.parallel to Y-axis.(x& z)
3.parallel to Z-axis.(x& y)
 =>1

ENter Rotation angle: 45
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

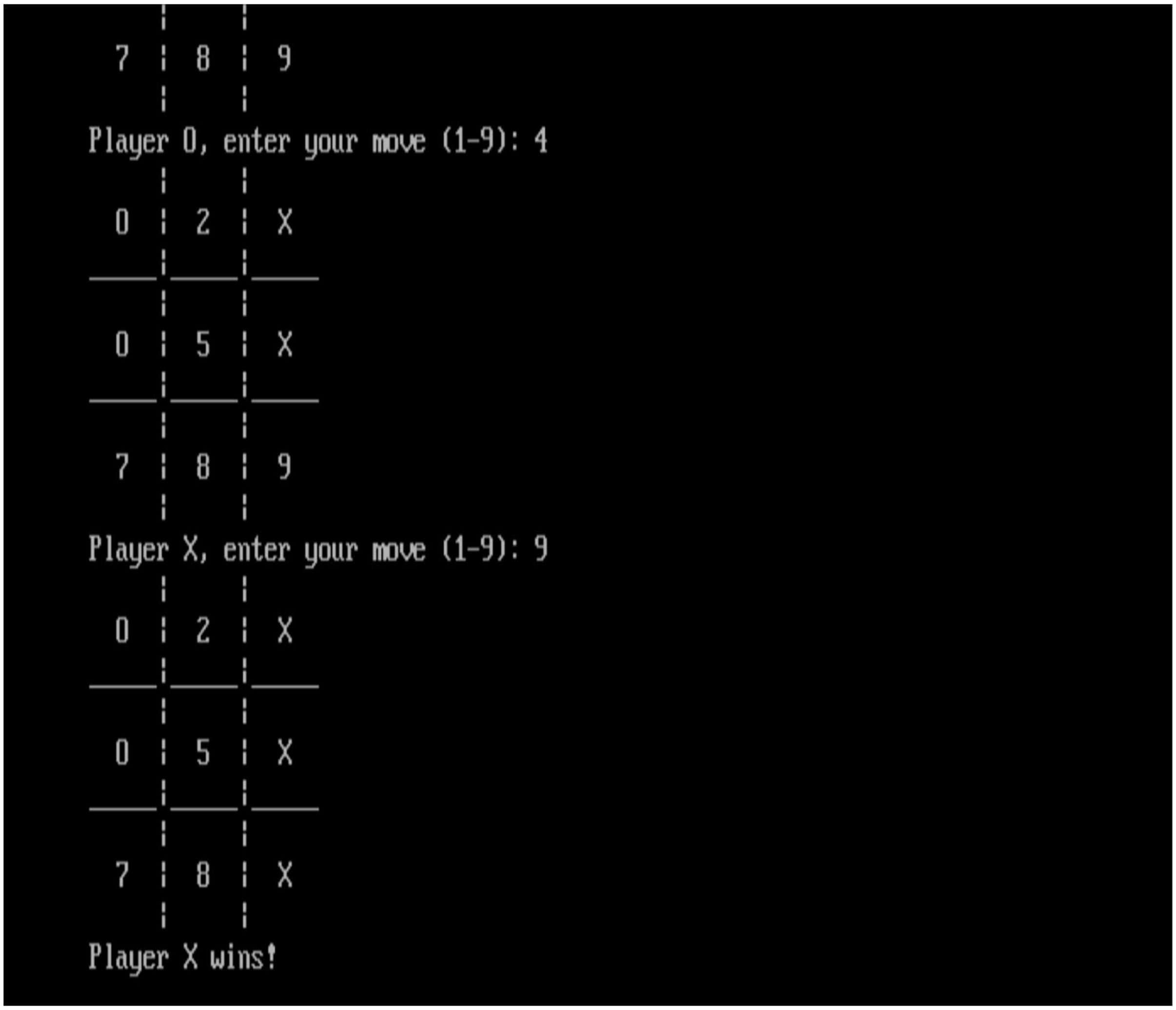
Experiment no 10

```
#include <iostream>  // Include iostream for input/output
#include <conio.h>   // Include conio.h for getch()
#include <dos.h>     // Include dos.h for delay (optional)

using namespace std;

// Global variable to represent the Tic-Tac-Toe board
char board[3][3] = { {'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'} };

// Function prototypes
void displayBoard();
bool checkWin();
bool checkDraw();
void makeMove(char player);

int main() {
    char player = 'X';  // Starting player
    bool gameWon = false;  // Flag to check if the game is won
    bool draw = false;     // Flag to check if the game is a draw

    // Game loop
    while (!gameWon && !draw) {
        displayBoard();         // Display the current board
        makeMove(player);      // Current player makes a move

        gameWon = checkWin();  // Check if the current player has won
        draw = checkDraw();     // Check if the game is a draw

        // Switch player if the game is not won or drawn
        if (!gameWon && !draw) {
            player = (player == 'X') ? 'O' : 'X'; // Toggle between 'X' and 'O'
        }
    }

    displayBoard();  // Display the final board

    // Announce the result
    if (gameWon) {
        cout << "Player " << player << " wins!" << endl;
    } else {
        cout << "It's a draw!" << endl;
    }

    getch();  // Wait for a key press before closing
    return 0;  // End of the program
}

// Function to display the current board
void displayBoard() {
    cout << "     |     |     " << endl;
    cout << "  " << board[0][0] << "  |  " << board[0][1] << "  |  " << board[0][2] << endl;
    cout << "_____|_____|_____" << endl;
    cout << "     |     |     " << endl;
    cout << "  " << board[1][0] << "  |  " << board[1][1] << "  |  " << board[1][2] << endl;
    cout << "_____|_____|_____" << endl;
    cout << "     |     |     " << endl;
    cout << "  " << board[2][0] << "  |  " << board[2][1] << "  |  " << board[2][2] << endl;
    cout << "     |     |     " << endl;
}

// Function to check if any player has won
bool checkWin() {
    // Check rows and columns
    for (int i = 0; i < 3; i++) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2])
            return true;  // Row win
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i])
            return true;  // Column win
    }

    // Check diagonals
    if (board[0][0] == board[1][1] && board[1][1] == board[2][2])
        return true;  // Diagonal win
    if (board[0][2] == board[1][1] && board[1][1] == board[2][0])
        return true;  // Diagonal win

    return false;  // No win
}

// Function to check if the game is a draw
bool checkDraw() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[i][j] != 'X' && board[i][j] != 'O') {
                return false;  // There are still moves left
            }
        }
    }
    return true;  // No moves left, it's a draw
}

// Function to make a move for the current player
void makeMove(char player) {
    int choice;
    cout << "Player " << player << ", enter your move (1-9): ";
    cin >> choice;

    int row = (choice - 1) / 3;  // Calculate row index
    int col = (choice - 1) % 3;  // Calculate column index

    // Check if the chosen spot is valid
    if (choice >= 1 && choice <= 9 && board[row][col] != 'X' && board[row][col] != 'O') {
        board[row][col] = player;  // Update the board
    } else {
        cout << "Invalid move! Try again." << endl;
        makeMove(player);  // Recursively prompt for a valid move
    }
}
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)

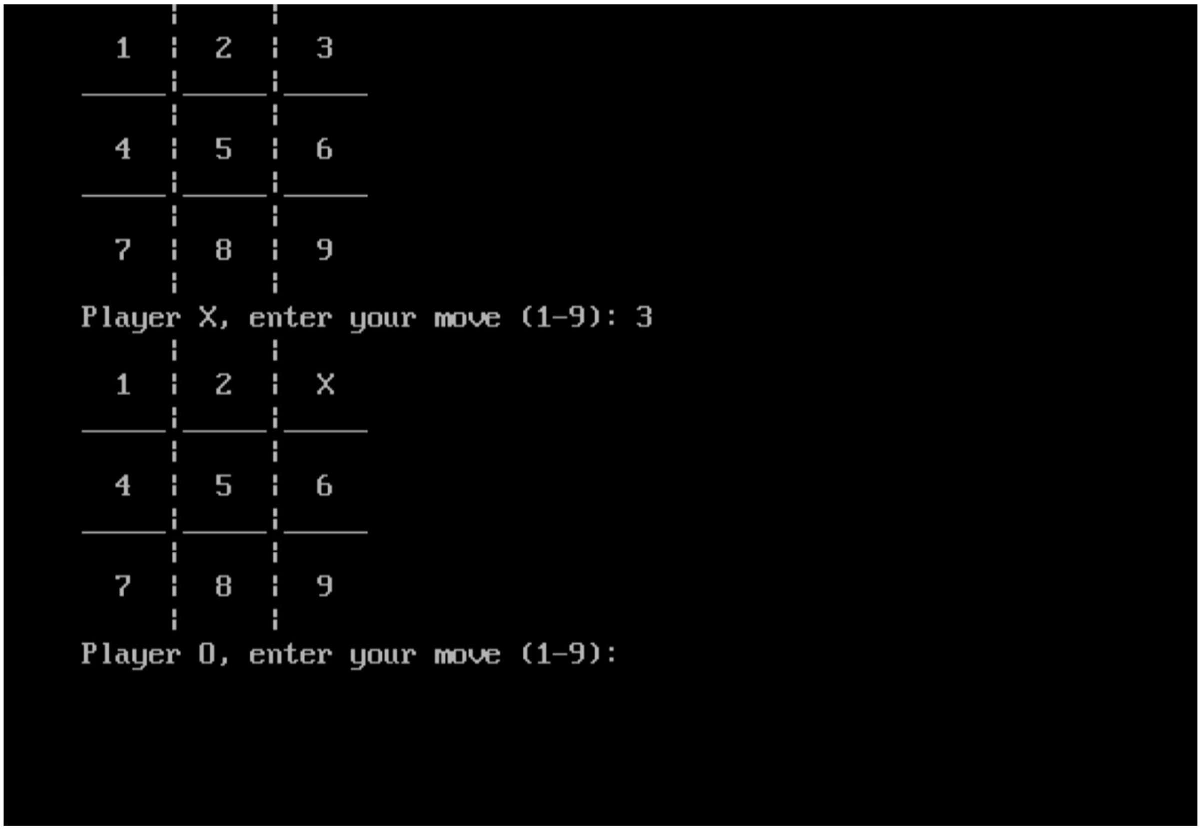


```
     ¦     ¦
  7  ¦  8  ¦  9
     ¦     ¦
Player O, enter your move (1-9): 4
     ¦     ¦
  O  ¦  2  ¦  X
_____¦_____¦_____
     ¦     ¦
  O  ¦  5  ¦  X
_____¦_____¦_____
     ¦     ¦
  7  ¦  8  ¦  9
     ¦     ¦
Player X, enter your move (1-9): 9
     ¦     ¦
  O  ¦  2  ¦  X
_____¦_____¦_____
     ¦     ¦
  O  ¦  5  ¦  X
_____¦_____¦_____
     ¦     ¦
  7  ¦  8  ¦  X
     ¦     ¦
Player X wins!
```

Name: Bhupen Jitendra Chirmade

Roll Number: 22

Class: SE Computer (A)



```
       |     |
   1   |  2  |  3
 ______|_____|______
       |     |
   4   |  5  |  6
 ______|_____|______
       |     |
   7   |  8  |  9
       |     |
 Player X, enter your move (1-9): 3
       |     |
   1   |  2  |  X
 ______|_____|______
       |     |
   4   |  5  |  6
 ______|_____|______
       |     |
   7   |  8  |  9
       |     |
 Player O, enter your move (1-9):
```