Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Computer Graphics(COMP 342) – Lab 3

Submitted To:

Dhiraj Shrestha Sir

Department of Computer Science and Engineering

Submitted By:

Mission Shrestha (54)

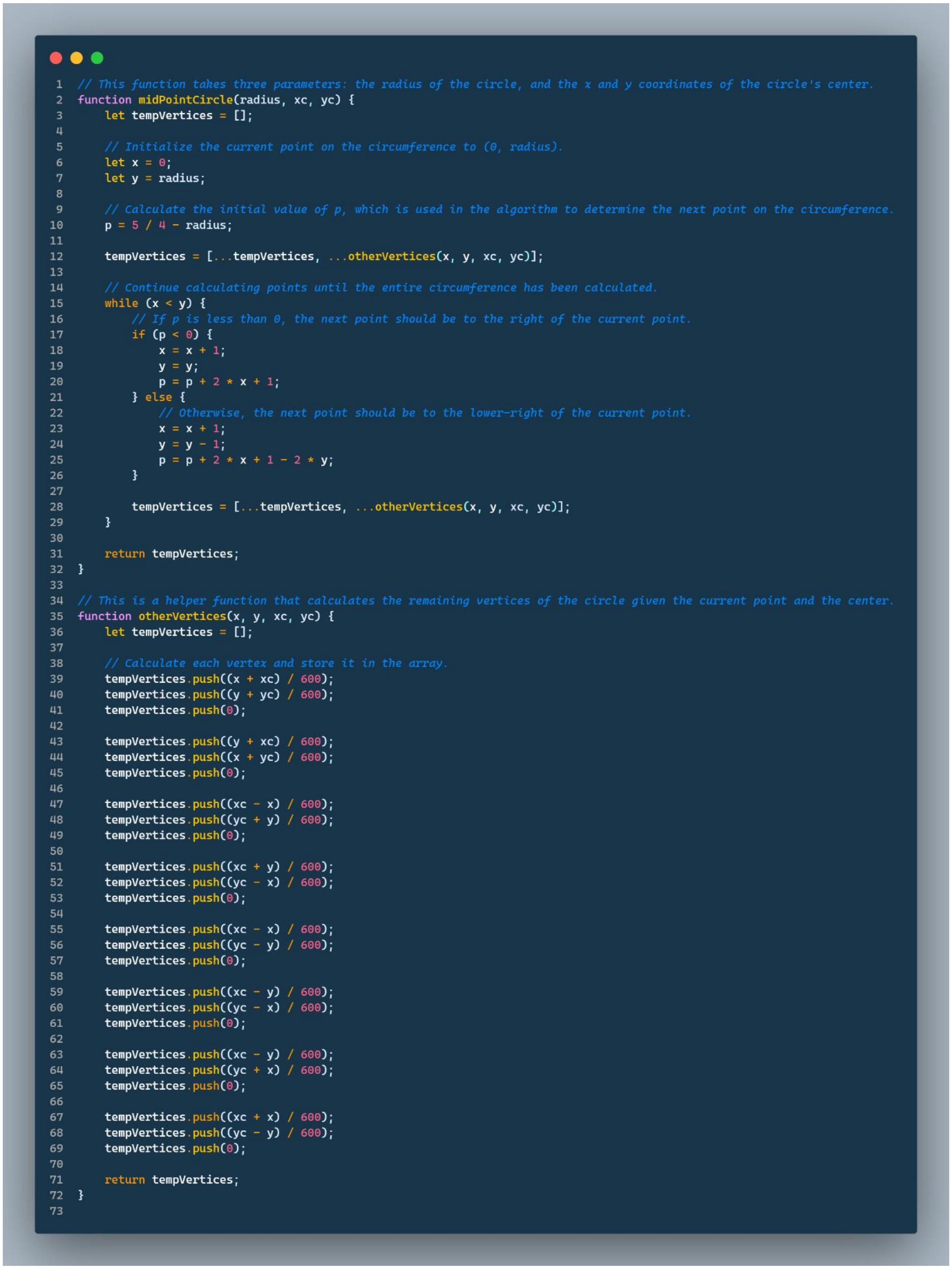
CE-2019 Year/Sem: 3rd Year/2nd Sem

Submission Date: 26th April, 2023

Implementation of mid- point Circle & Ellipse Drawing Algorithm.

1. Circle Algorithm
Source Code:

```
// This function takes three parameters: the radius of the circle, and the x and y
coordinates of the circle's center.
function midPointCircle(radius, xc, yc) {
    let tempVertices = [];

    // Initialize the current point on the circumference to (0, radius).
    let x = 0;
    let y = radius;

    // Calculate the initial value of p, which is used in the algorithm to
determine the next point on the circumference.
    p = 5 / 4 - radius;

    tempVertices = [...tempVertices, ...otherVertices(x, y, xc, yc)];

    // Continue calculating points until the entire circumference has been
calculated.
    while (x < y) {
        // If p is less than 0, the next point should be to the right of the
current point.
        if (p < 0) {
            x = x + 1;
            y = y;
            p = p + 2 * x + 1;
        } else {
            // Otherwise, the next point should be to the lower-right of the
current point.
            x = x + 1;
            y = y - 1;
            p = p + 2 * x + 1 - 2 * y;
        }

        tempVertices = [...tempVertices, ...otherVertices(x, y, xc, yc)];
    }

    return tempVertices;
}

// This is a helper function that calculates the remaining vertices of the circle
given the current point and the center.
function otherVertices(x, y, xc, yc) {
    let tempVertices = [];

    // Calculate each vertex and store it in the array.
    tempVertices.push((x + xc) / 600);
    tempVertices.push((y + yc) / 600);
    tempVertices.push(0);

    tempVertices.push((y + xc) / 600);
    tempVertices.push((x + yc) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - x) / 600);
    tempVertices.push((yc + y) / 600);
    tempVertices.push(0);

    tempVertices.push((xc + y) / 600);
    tempVertices.push((yc - x) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - x) / 600);
    tempVertices.push((yc - y) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - y) / 600);
    tempVertices.push((yc - x) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - y) / 600);
    tempVertices.push((yc + x) / 600);
    tempVertices.push(0);

    tempVertices.push((xc + x) / 600);
    tempVertices.push((yc - y) / 600);
    tempVertices.push(0);

    return tempVertices;
}
```

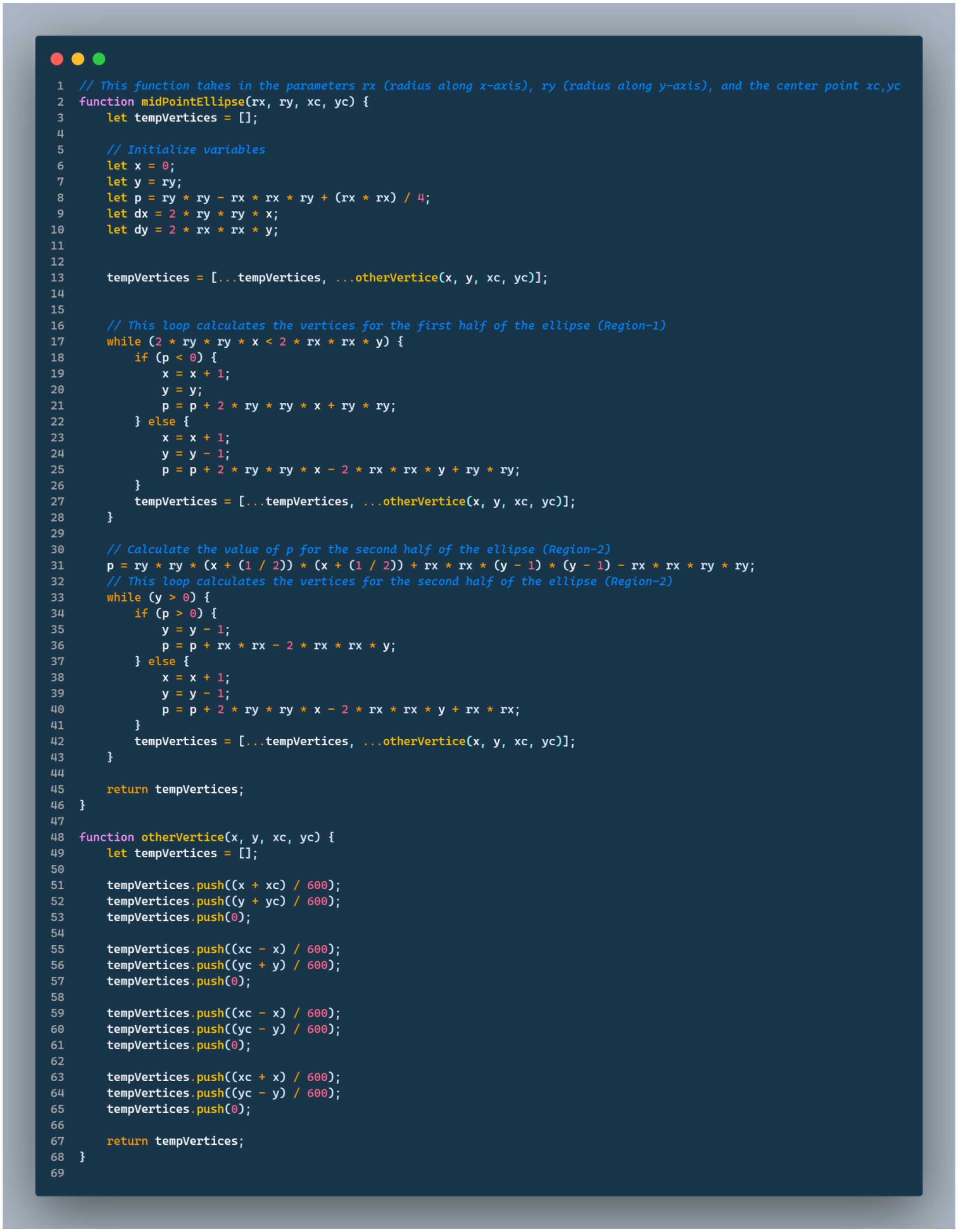


```javascript
// This function takes three parameters: the radius of the circle, and the x and y coordinates of the circle's center.
function midPointCircle(radius, xc, yc) {
    let tempVertices = [];

    // Initialize the current point on the circumference to (0, radius).
    let x = 0;
    let y = radius;

    // Calculate the initial value of p, which is used in the algorithm to determine the next point on the circumference.
    p = 5 / 4 - radius;

    tempVertices = [...tempVertices, ...otherVertices(x, y, xc, yc)];

    // Continue calculating points until the entire circumference has been calculated.
    while (x < y) {
        // If p is less than 0, the next point should be to the right of the current point.
        if (p < 0) {
            x = x + 1;
            y = y;
            p = p + 2 * x + 1;
        } else {
            // Otherwise, the next point should be to the lower-right of the current point.
            x = x + 1;
            y = y - 1;
            p = p + 2 * x + 1 - 2 * y;
        }

        tempVertices = [...tempVertices, ...otherVertices(x, y, xc, yc)];
    }

    return tempVertices;
}

// This is a helper function that calculates the remaining vertices of the circle given the current point and the center.
function otherVertices(x, y, xc, yc) {
    let tempVertices = [];

    // Calculate each vertex and store it in the array.
    tempVertices.push((x + xc) / 600);
    tempVertices.push((y + yc) / 600);
    tempVertices.push(0);

    tempVertices.push((y + xc) / 600);
    tempVertices.push((x + yc) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - x) / 600);
    tempVertices.push((yc + y) / 600);
    tempVertices.push(0);

    tempVertices.push((xc + y) / 600);
    tempVertices.push((yc - x) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - x) / 600);
    tempVertices.push((yc - y) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - y) / 600);
    tempVertices.push((yc - x) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - y) / 600);
    tempVertices.push((yc + x) / 600);
    tempVertices.push(0);

    tempVertices.push((xc + x) / 600);
    tempVertices.push((yc - y) / 600);
    tempVertices.push(0);

    return tempVertices;
}
```

2. Ellipse Algorithm

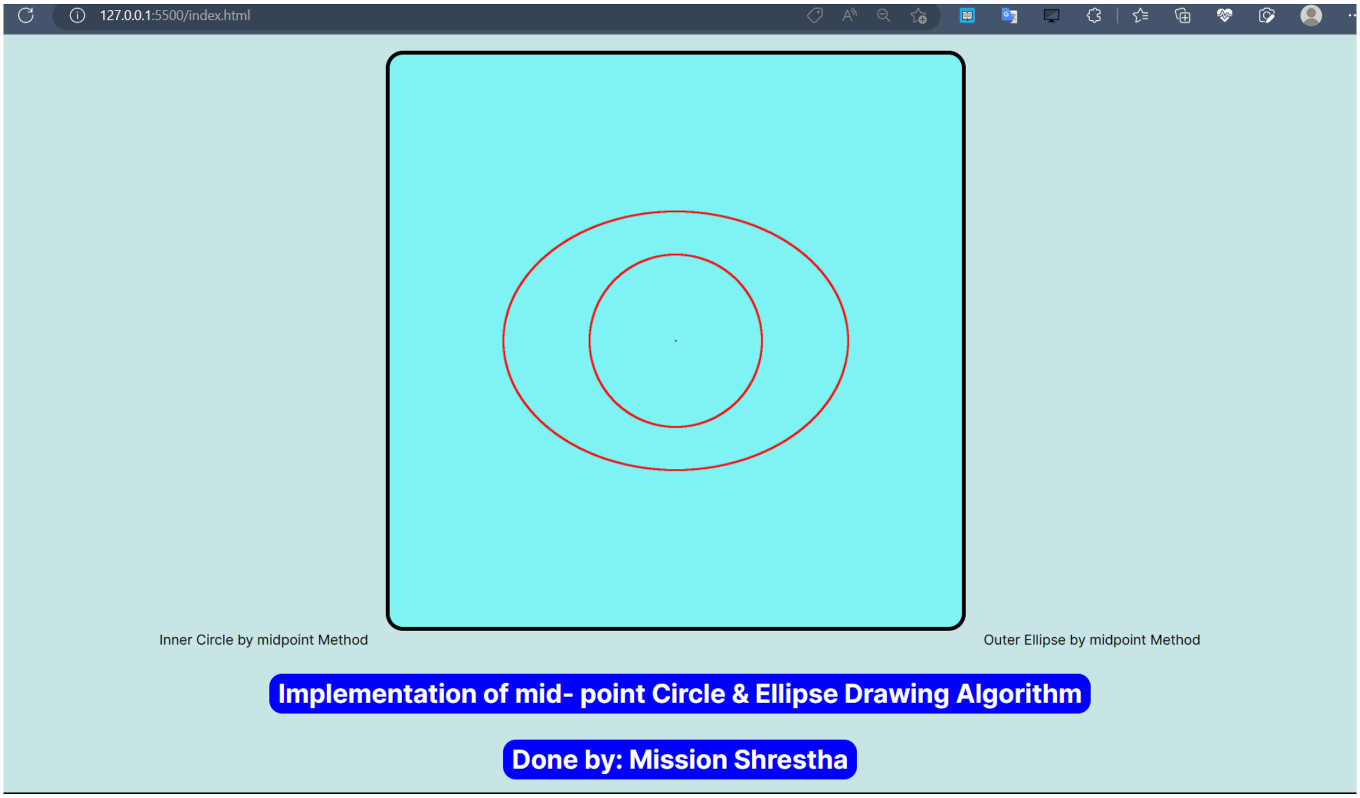
Source Code:

```
    // This function takes in the parameters rx (radius along x-axis), ry (radius along y-axis), and the center point xc,yc
    function midPointEllipse(rx, ry, xc, yc) {
        let tempVertices = [];

        // Initialize variables
        let x = 0;
        let y = ry;
        let p = ry * ry - rx * rx * ry + (rx * rx) / 4;
        let dx = 2 * ry * ry * x;
        let dy = 2 * rx * rx * y;

        tempVertices = [...tempVertices, ...otherVertice(x, y, xc, yc)];

        // This loop calculates the vertices for the first half of the ellipse (Region-1)
        while (2 * ry * ry * x < 2 * rx * rx * y) {
            if (p < 0) {
                x = x + 1;
                y = y;
                p = p + 2 * ry * ry * x + ry * ry;
            } else {
                x = x + 1;
                y = y - 1;
                p = p + 2 * ry * ry * x - 2 * rx * rx * y + ry * ry;
            }
            tempVertices = [...tempVertices, ...otherVertice(x, y, xc, yc)];
        }

        // Calculate the value of p for the second half of the ellipse (Region-2)
        p = ry * ry * (x + (1 / 2)) * (x + (1 / 2)) + rx * rx * (y - 1) * (y - 1) - rx * rx * ry * ry;
    // This loop calculates the vertices for the second half of the ellipse (Region-2)
    while (y > 0) {
        if (p > 0) {
            y = y - 1;
            p = p + rx * rx - 2 * rx * rx * y;
        } else {
            x = x + 1;
            y = y - 1;
            p = p + 2 * ry * ry * x - 2 * rx * rx * y + rx * rx;
        }
        tempVertices = [...tempVertices, ...otherVertice(x, y, xc, yc)];
    }

    return tempVertices;
}

function otherVertice(x, y, xc, yc) {
    let tempVertices = [];

    tempVertices.push((x + xc) / 600);
    tempVertices.push((y + yc) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - x) / 600);
    tempVertices.push((yc + y) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - x) / 600);
    tempVertices.push((yc - y) / 600);
    tempVertices.push(0);

    tempVertices.push((xc + x) / 600);
    tempVertices.push((yc - y) / 600);
    tempVertices.push(0);

    return tempVertices;
}

```



```
// This function takes in the parameters rx (radius along x-axis), ry (radius along y-axis), and the center point xc,yc
function midPointEllipse(rx, ry, xc, yc) {
    let tempVertices = [];

    // Initialize variables
    let x = 0;
    let y = ry;
    let p = ry * ry - rx * rx * ry + (rx * rx) / 4;
    let dx = 2 * ry * ry * x;
    let dy = 2 * rx * rx * y;


    tempVertices = [...tempVertices, ...otherVertice(x, y, xc, yc)];


    // This loop calculates the vertices for the first half of the ellipse (Region-1)
    while (2 * ry * ry * x < 2 * rx * rx * y) {
        if (p < 0) {
            x = x + 1;
            y = y;
            p = p + 2 * ry * ry * x + ry * ry;
        } else {
            x = x + 1;
            y = y - 1;
            p = p + 2 * ry * ry * x - 2 * rx * rx * y + ry * ry;
        }
        tempVertices = [...tempVertices, ...otherVertice(x, y, xc, yc)];
    }

    // Calculate the value of p for the second half of the ellipse (Region-2)
    p = ry * ry * (x + (1 / 2)) * (x + (1 / 2)) + rx * rx * (y - 1) * (y - 1) - rx * rx * ry * ry;
    // This loop calculates the vertices for the second half of the ellipse (Region-2)
    while (y > 0) {
        if (p > 0) {
            y = y - 1;
            p = p + rx * rx - 2 * rx * rx * y;
        } else {
            x = x + 1;
            y = y - 1;
            p = p + 2 * ry * ry * x - 2 * rx * rx * y + rx * rx;
        }
        tempVertices = [...tempVertices, ...otherVertice(x, y, xc, yc)];
    }

    return tempVertices;
}

function otherVertice(x, y, xc, yc) {
    let tempVertices = [];

    tempVertices.push((x + xc) / 600);
    tempVertices.push((y + yc) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - x) / 600);
    tempVertices.push((yc + y) / 600);
    tempVertices.push(0);

    tempVertices.push((xc - x) / 600);
    tempVertices.push((yc - y) / 600);
    tempVertices.push(0);

    tempVertices.push((xc + x) / 600);
    tempVertices.push((yc - y) / 600);
    tempVertices.push(0);

    return tempVertices;
}
```

Output:



Inner Circle by midpoint Method

Outer Ellipse by midpoint Method

Implementation of mid- point Circle & Ellipse Drawing Algorithm

Done by: Mission Shrestha

Conclusion:

Hence, midpoint algorithm was used to draw circle & ellipse.