WORKSHEET 6

Student Name: Sumit Kumar **UID: 22BCS10048**

Section/Group: KRG_IOT-1-A **Semester: 6th Semester**

Branch: BE-CSE **Date of Performance: 13/02/2025**

Subject Name: Computer Graphics with Lab **Subject Code: 22CSH-352**

1. Aim: Analyze and implement the reflection of a point about a line defined by the equation y = mx + c.

2. Objective: To implement and analyze the reflection of a point about a straight line defined by the equation y = mx + c.

3. Implementation Code:

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

using namespace std;

void reflectPoint(float x, float y, float m, float c, float &refX, float &refY) {
    float d = (x + (y - c) * m) / (1 + m * m);
    refX = 2 * d - x;
    refY = 2 * d * m - y + 2 * c;
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    float x, y, m, c, refX, refY;

    cout << "Enter coordinates of the point (x, y): ";
    cin >> x >> y;
    cout << "Enter slope (m) and y-intercept (c) of the line (y = mx + c): ";
    cin >> m >> c;

    reflectPoint(x, y, m, c, refX, refY);

    float x1 = 0, y1 = c;
    float x2 = getmaxx(), y2 = m * x2 + c;
    setcolor(WHITE);
    line(x1, y1, x2, y2);

    setcolor(WHITE);
    circle(x, y, 5);
    outtextxy(x + 10, y, "Original");
```

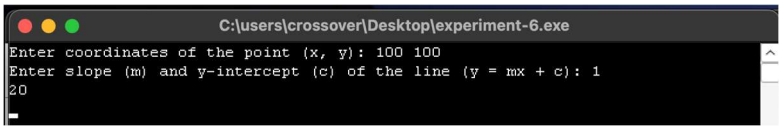
```
    setcolor(WHITE);
    circle(refX, refY, 5);
    outtextxy(refX + 10, refY, "Reflected");
    getch();
    closegraph();
    return 0;
}
```

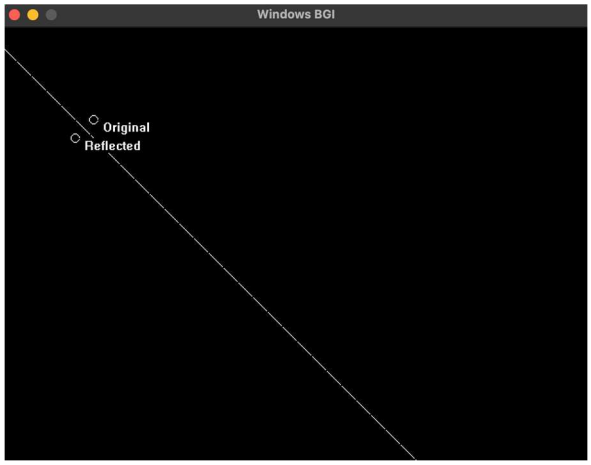
4. Explanation:

This program calculates and plots the reflection of a point across a given line defined by the equation y= mx + c. The reflection formula is derived from the perpendicular distance concept. The program first takes user inputs for the point coordinates and the line equation parameters (slope m and intercept c). It then computes the reflected point using mathematical transformations and plots both the original and reflected points on the graphical interface.

This visualization helps in understanding the transformation's correctness. The implementation demonstrates how geometry and algebra are used to manipulate graphical objects, making it a useful application of computer graphics in simulations and CAD software.

5. Output:





6. Learning Outcomes:

- Understanding 2D rotation transformations in computer graphics.
- Applying trigonometric functions for accurate geometric transformations.
- Implementing clockwise and anticlockwise rotations using matrices.
- Using `graphics.h` to visualize transformations dynamically.
- Evaluating the impact of pivot-based transformations on object positioning.