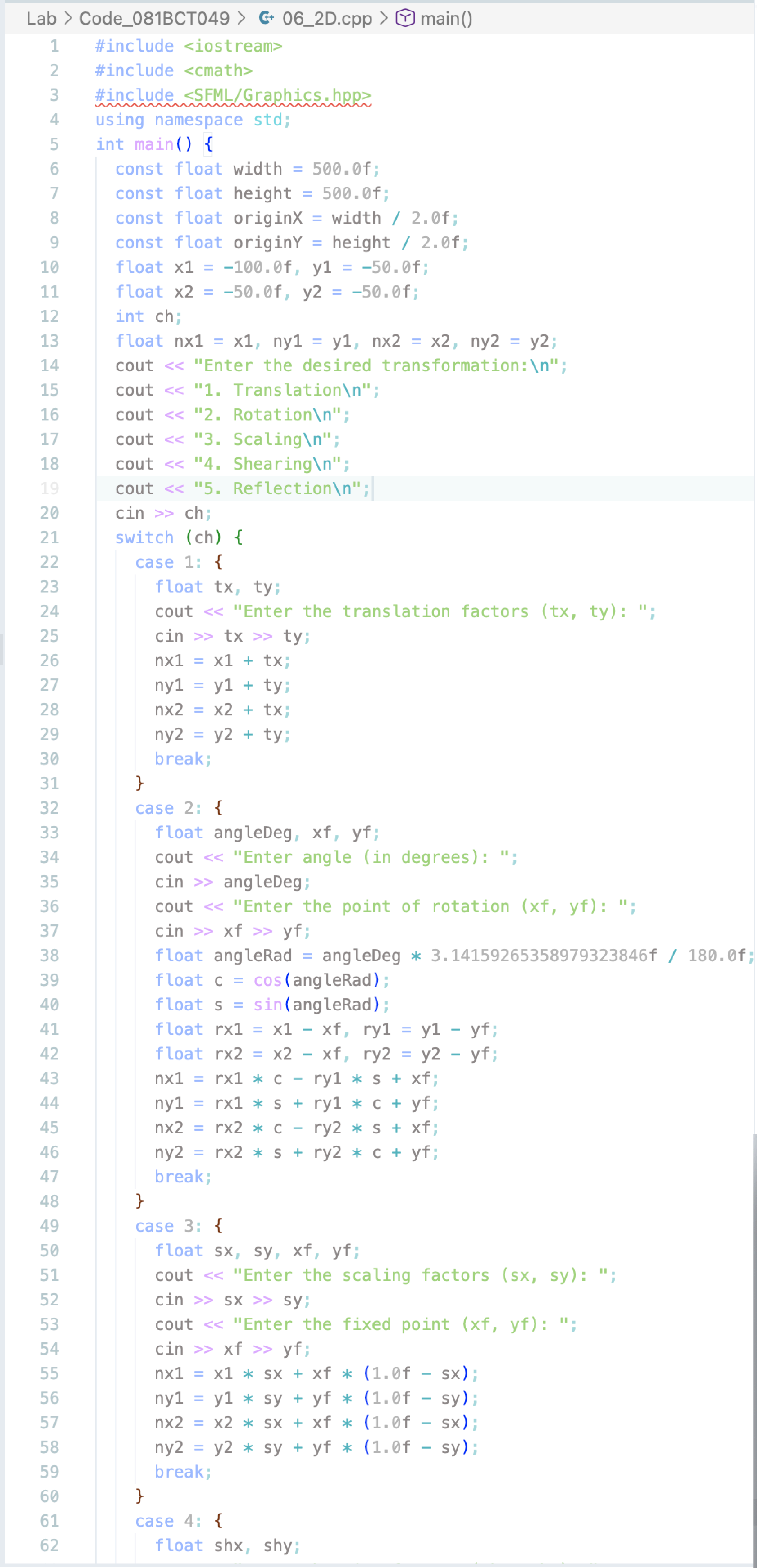
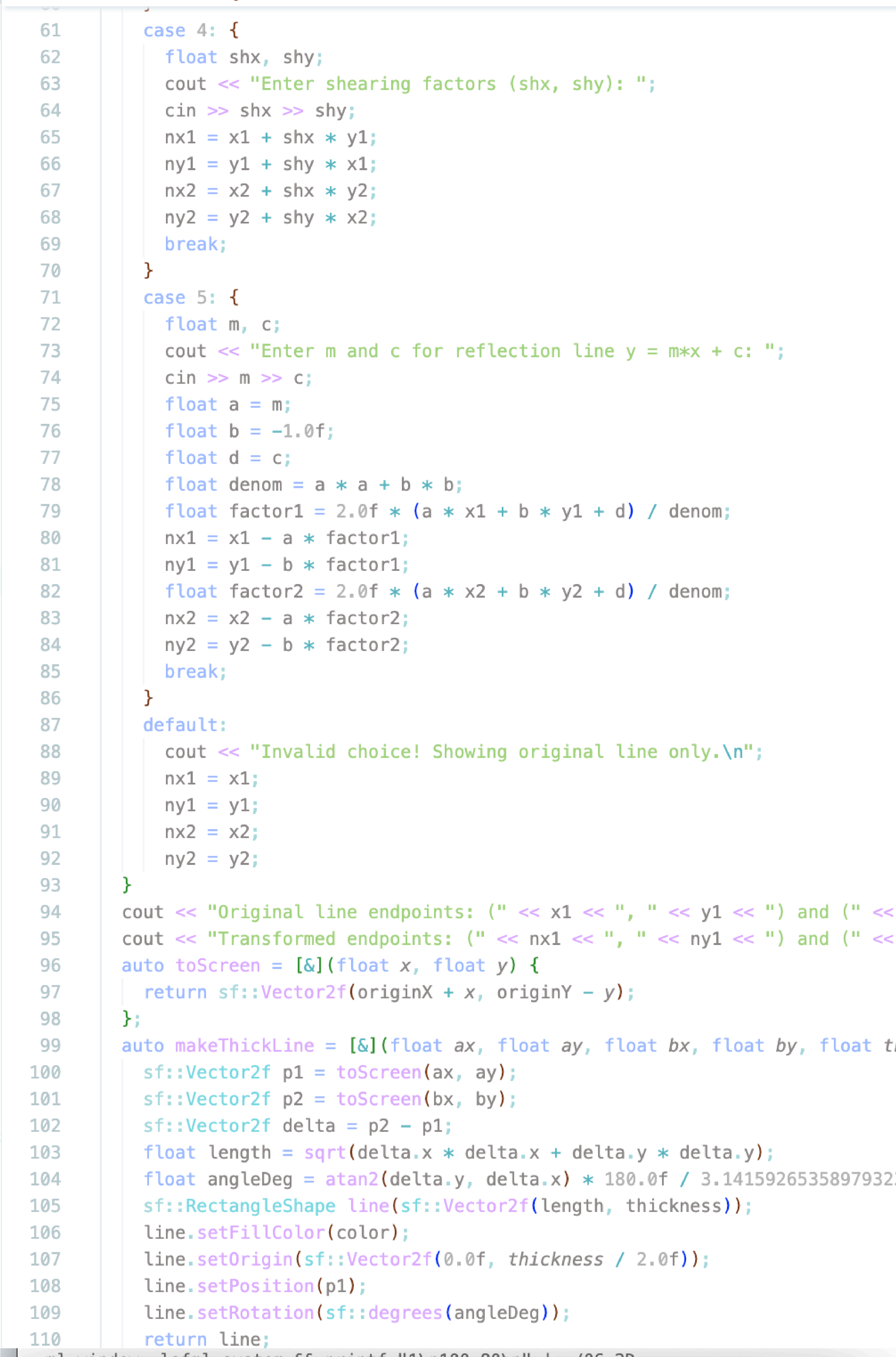


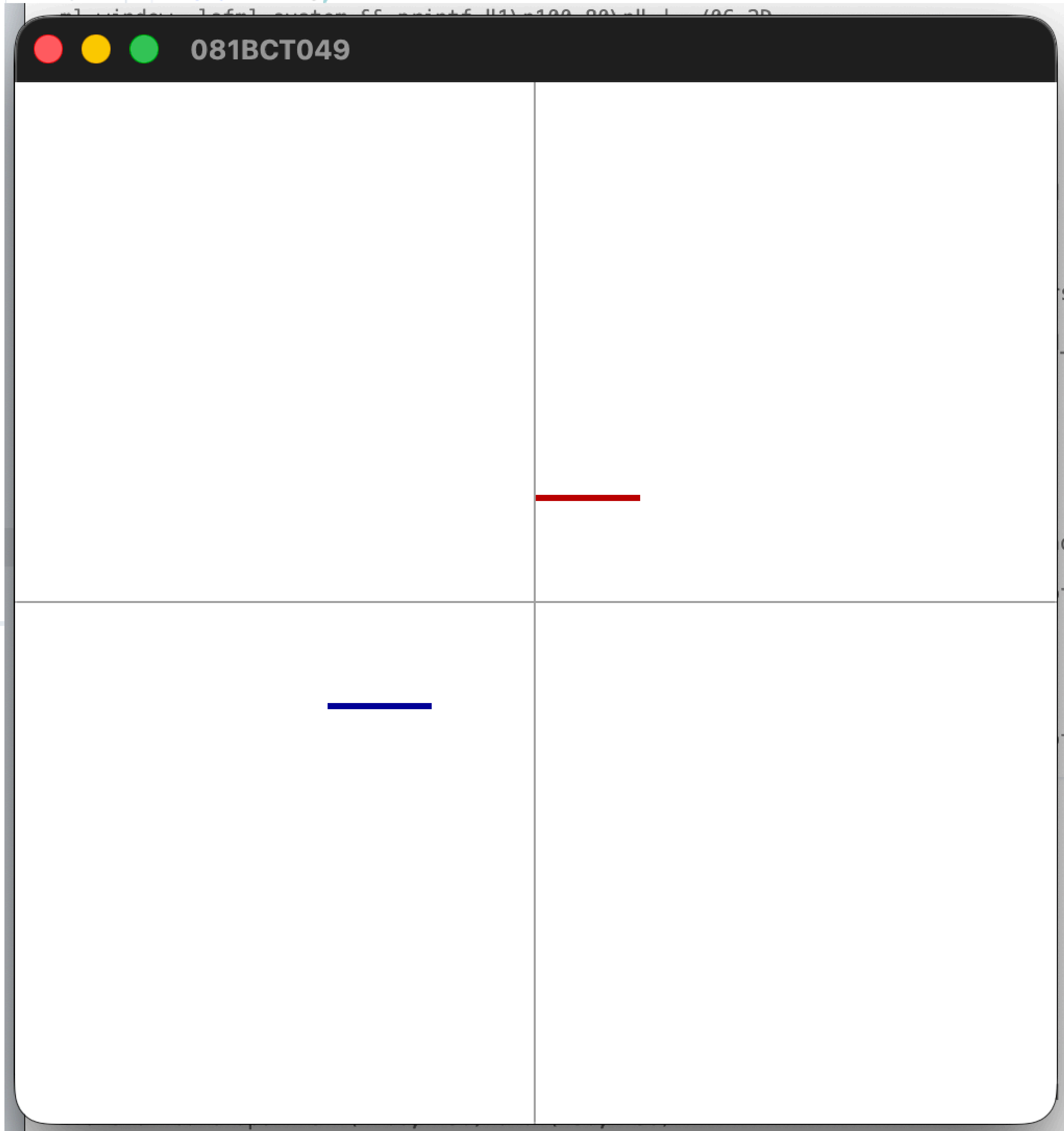
Lab > Code_081BCT049 > 06_2D.cpp > main()

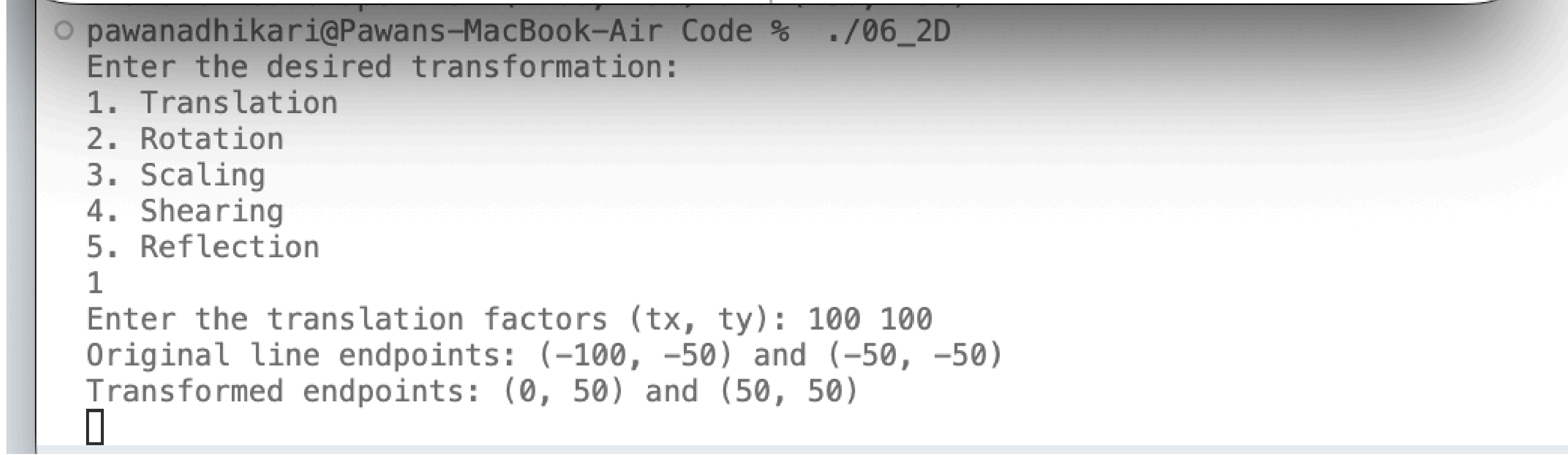
```cpp
#include <iostream>
#include <cmath>
#include <SFML/Graphics.hpp>
using namespace std;
int main() {
  const float width = 500.0f;
  const float height = 500.0f;
  const float originX = width / 2.0f;
  const float originY = height / 2.0f;
  float x1 = -100.0f, y1 = -50.0f;
  float x2 = -50.0f, y2 = -50.0f;
  int ch;
  float nx1 = x1, ny1 = y1, nx2 = x2, ny2 = y2;
  cout << "Enter the desired transformation:\n";
  cout << "1. Translation\n";
  cout << "2. Rotation\n";
  cout << "3. Scaling\n";
  cout << "4. Shearing\n";
  cout << "5. Reflection\n";
  cin >> ch;
  switch (ch) {
    case 1: {
      float tx, ty;
      cout << "Enter the translation factors (tx, ty): ";
      cin >> tx >> ty;
      nx1 = x1 + tx;
      ny1 = y1 + ty;
      nx2 = x2 + tx;
      ny2 = y2 + ty;
      break;
    }
    case 2: {
      float angleDeg, xf, yf;
      cout << "Enter angle (in degrees): ";
      cin >> angleDeg;
      cout << "Enter the point of rotation (xf, yf): ";
      cin >> xf >> yf;
      float angleRad = angleDeg * 3.14159265358979323846f / 180.0f;
      float c = cos(angleRad);
      float s = sin(angleRad);
      float rx1 = x1 - xf, ry1 = y1 - yf;
      float rx2 = x2 - xf, ry2 = y2 - yf;
      nx1 = rx1 * c - ry1 * s + xf;
      ny1 = rx1 * s + ry1 * c + yf;
      nx2 = rx2 * c - ry2 * s + xf;
      ny2 = rx2 * s + ry2 * c + yf;
      break;
    }
    case 3: {
      float sx, sy, xf, yf;
      cout << "Enter the scaling factors (sx, sy): ";
      cin >> sx >> sy;
      cout << "Enter the fixed point (xf, yf): ";
      cin >> xf >> yf;
      nx1 = x1 * sx + xf * (1.0f - sx);
      ny1 = y1 * sy + yf * (1.0f - sy);
      nx2 = x2 * sx + xf * (1.0f - sx);
      ny2 = y2 * sy + yf * (1.0f - sy);
      break;
    }
    case 4: {
      float shx, shy;
```



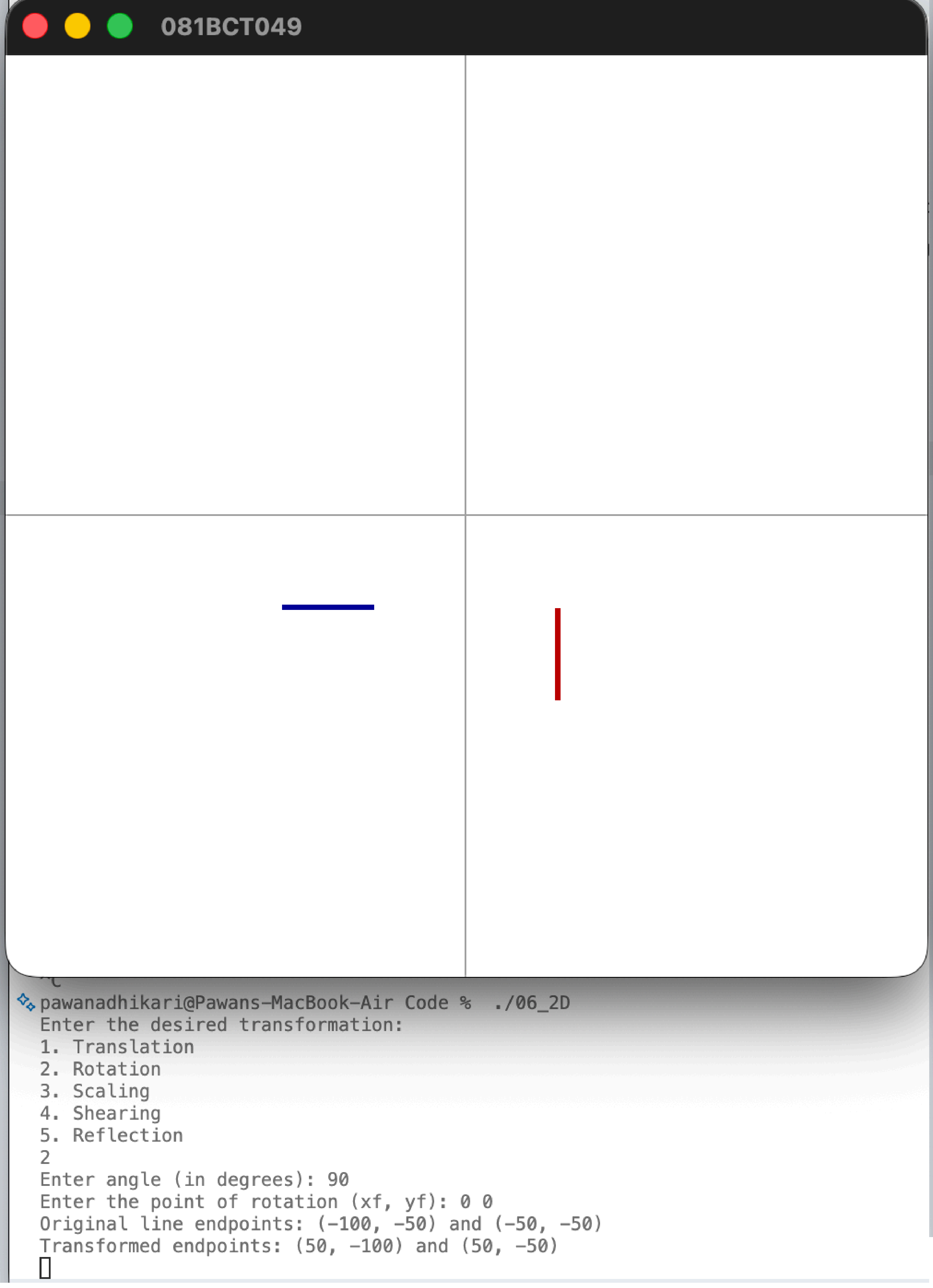
```cpp
    case 4: {
      float shx, shy;
```

```cpp
      cout << "Enter shearing factors (shx, shy): ";
      cin >> shx >> shy;
      nx1 = x1 + shx * y1;
      ny1 = y1 + shy * x1;
      nx2 = x2 + shx * y2;
      ny2 = y2 + shy * x2;
      break;
    }
    case 5: {
      float m, c;
      cout << "Enter m and c for reflection line y = m*x + c: ";
      cin >> m >> c;
      float a = m;
      float b = -1.0f;
      float d = c;
      float denom = a * a + b * b;
      float factor1 = 2.0f * (a * x1 + b * y1 + d) / denom;
      nx1 = x1 - a * factor1;
      ny1 = y1 - b * factor1;
      float factor2 = 2.0f * (a * x2 + b * y2 + d) / denom;
      nx2 = x2 - a * factor2;
      ny2 = y2 - b * factor2;
      break;
    }
    default:
      cout << "Invalid choice! Showing original line only.\n";
      nx1 = x1;
      ny1 = y1;
      nx2 = x2;
      ny2 = y2;
  }
  cout << "Original line endpoints: (" << x1 << ", " << y1 << ") and (" <<
  cout << "Transformed endpoints: (" << nx1 << ", " << ny1 << ") and (" <<
  auto toScreen = [&](float x, float y) {
    return sf::Vector2f(originX + x, originY - y);
  };
  auto makeThickLine = [&](float ax, float ay, float bx, float by, float t
    sf::Vector2f p1 = toScreen(ax, ay);
    sf::Vector2f p2 = toScreen(bx, by);
    sf::Vector2f delta = p2 - p1;
    float length = sqrt(delta.x * delta.x + delta.y * delta.y);
    float angleDeg = atan2(delta.y, delta.x) * 180.0f / 3.1415926535897932
    sf::RectangleShape line(sf::Vector2f(length, thickness));
    line.setFillColor(color);
    line.setOrigin(sf::Vector2f(0.0f, thickness / 2.0f));
    line.setPosition(p1);
    line.setRotation(sf::degrees(angleDeg));
    return line;
```

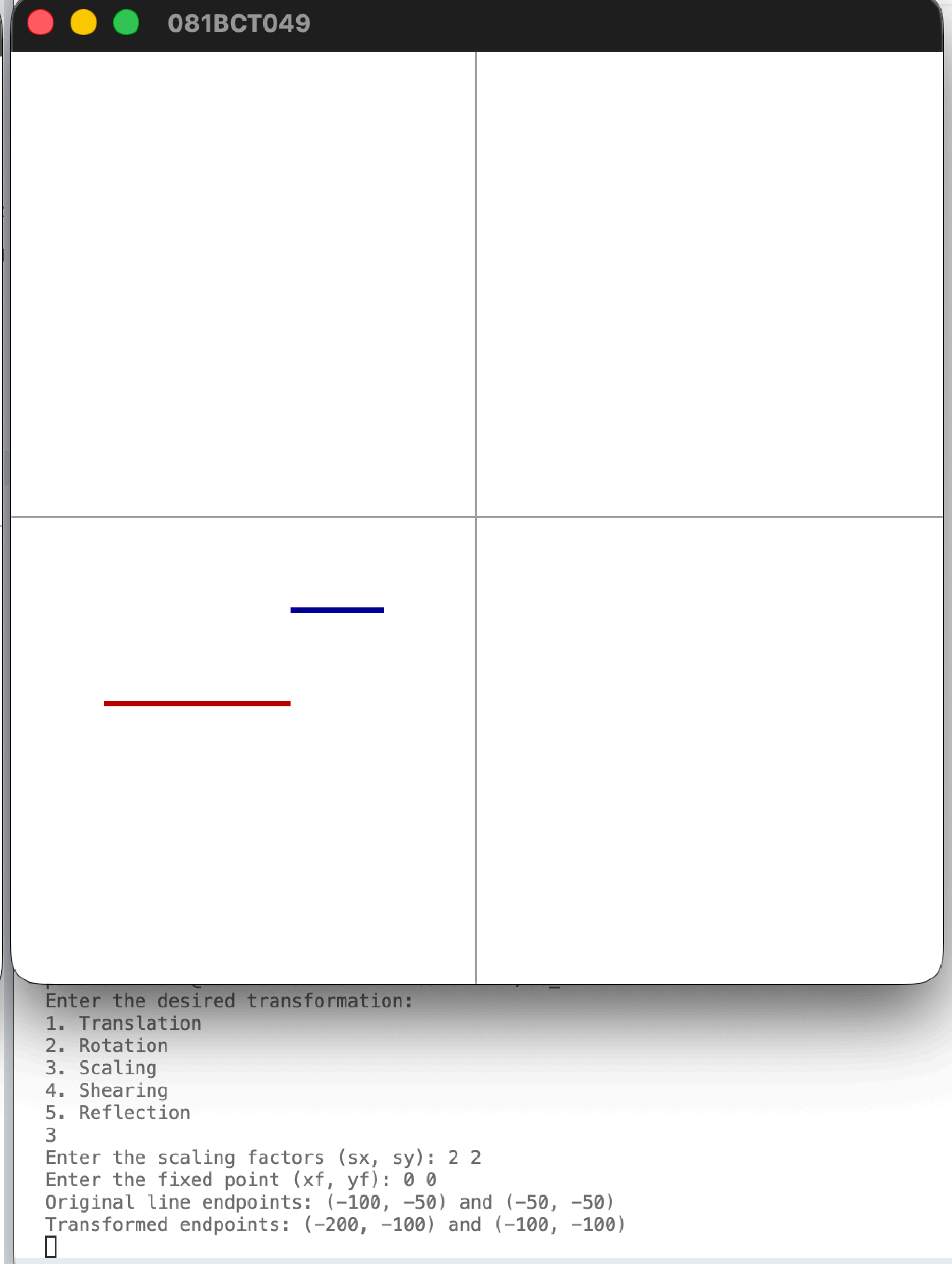




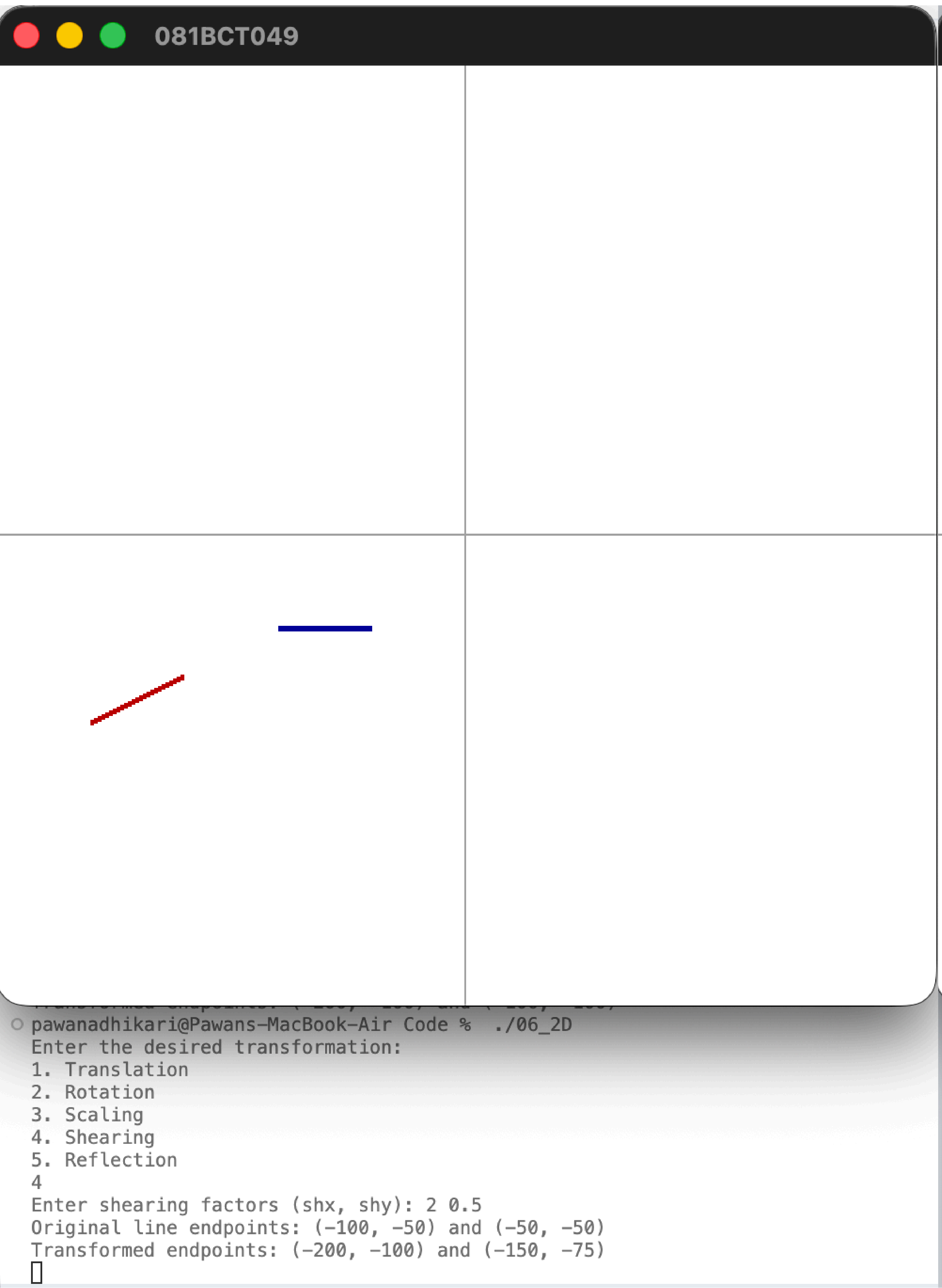
```
pawanadhikari@Pawans-MacBook-Air Code %  ./06_2D
Enter the desired transformation:
1. Translation
2. Rotation
3. Scaling
4. Shearing
5. Reflection
1
Enter the translation factors (tx, ty): 100 100
Original line endpoints: (-100, -50) and (-50, -50)
Transformed endpoints: (0, 50) and (50, 50)
```



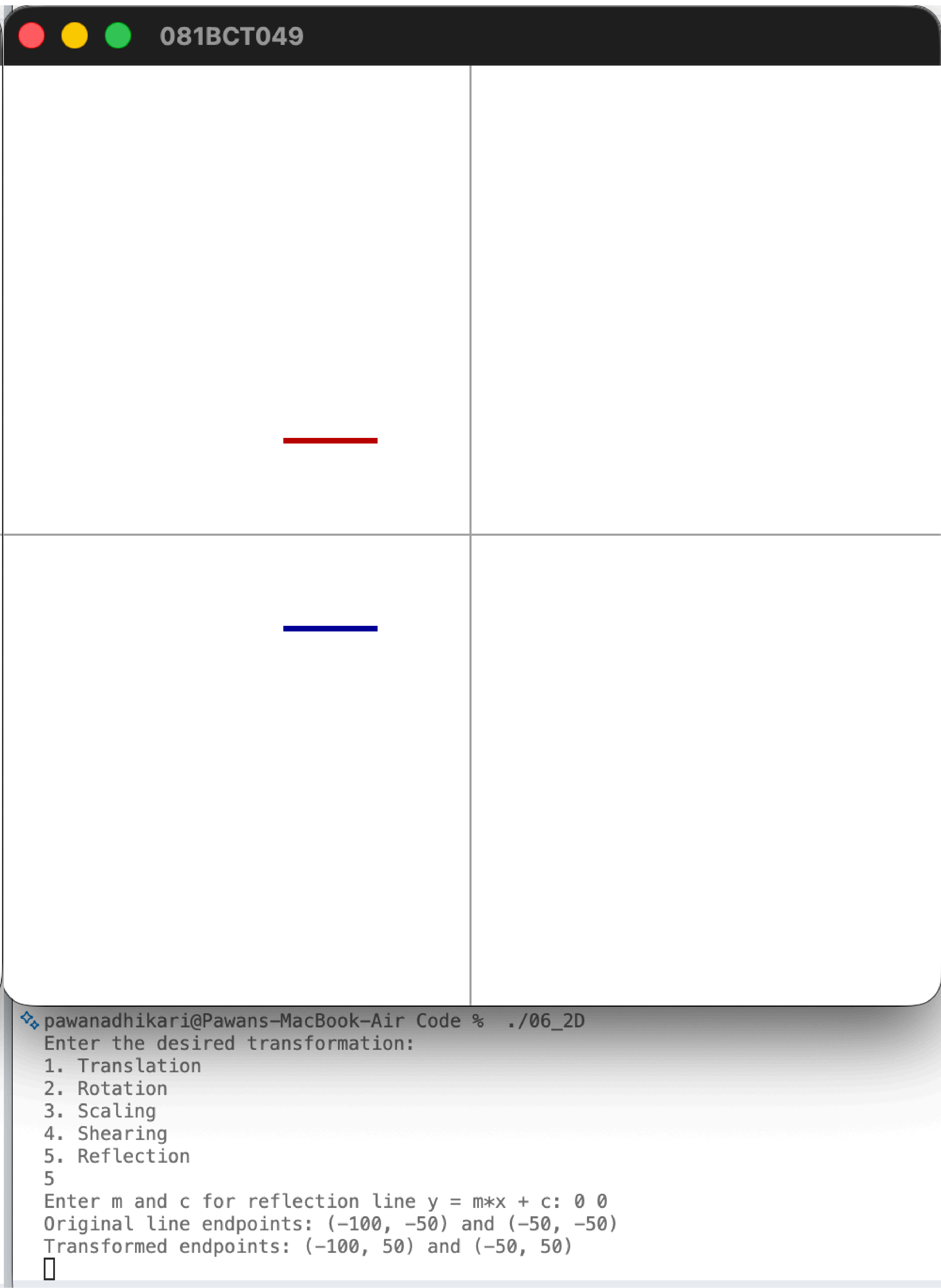
```
^C
pawanadhikari@Pawans-MacBook-Air Code %  ./06_2D
Enter the desired transformation:
1. Translation
2. Rotation
3. Scaling
4. Shearing
5. Reflection
2
Enter angle (in degrees): 90
Enter the point of rotation (xf, yf): 0 0
Original line endpoints: (-100, -50) and (-50, -50)
Transformed endpoints: (50, -100) and (50, -50)
```



```
Enter the desired transformation:
1. Translation
2. Rotation
3. Scaling
4. Shearing
5. Reflection
3
Enter the scaling factors (sx, sy): 2 2
Enter the fixed point (xf, yf): 0 0
Original line endpoints: (-100, -50) and (-50, -50)
Transformed endpoints: (-200, -100) and (-100, -100)
```



```
pawanadhikari@Pawans-MacBook-Air Code %  ./06_2D
Enter the desired transformation:
1. Translation
2. Rotation
3. Scaling
4. Shearing
5. Reflection
4
Enter shearing factors (shx, shy): 2 0.5
Original line endpoints: (-100, -50) and (-50, -50)
Transformed endpoints: (-200, -100) and (-150, -75)
```



```
pawanadhikari@Pawans-MacBook-Air Code %  ./06_2D
Enter the desired transformation:
1. Translation
2. Rotation
3. Scaling
4. Shearing
5. Reflection
5
Enter m and c for reflection line y = m*x + c: 0 0
Original line endpoints: (-100, -50) and (-50, -50)
Transformed endpoints: (-100, 50) and (-50, 50)
```