Stony Brook University
College of Engineering and Applied Science

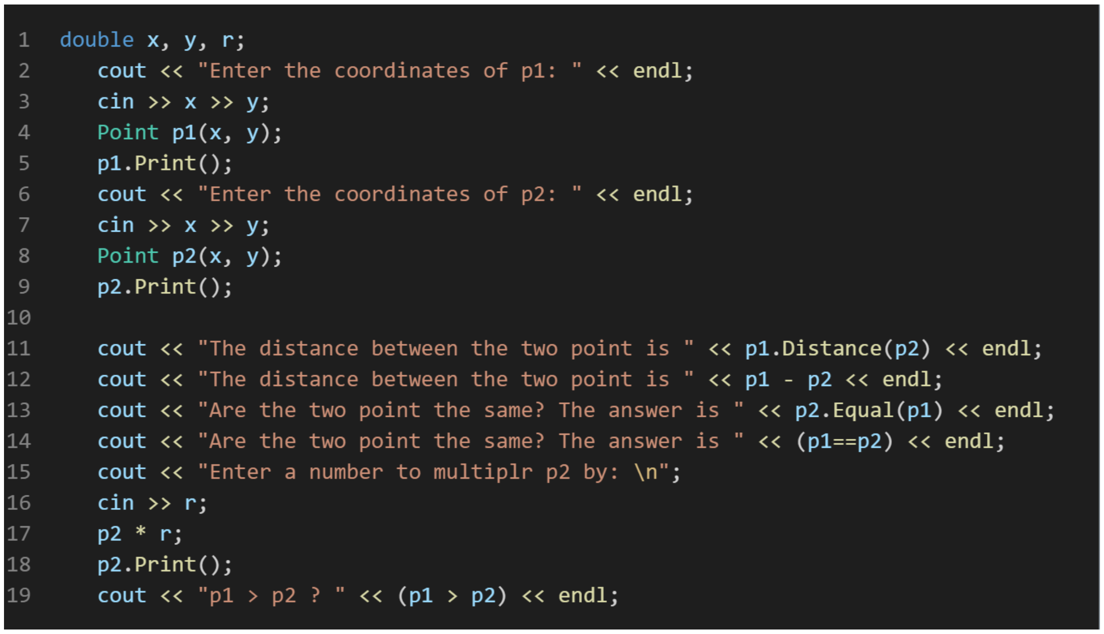
ESE 224.L02

Lab 2

Ryan Lin

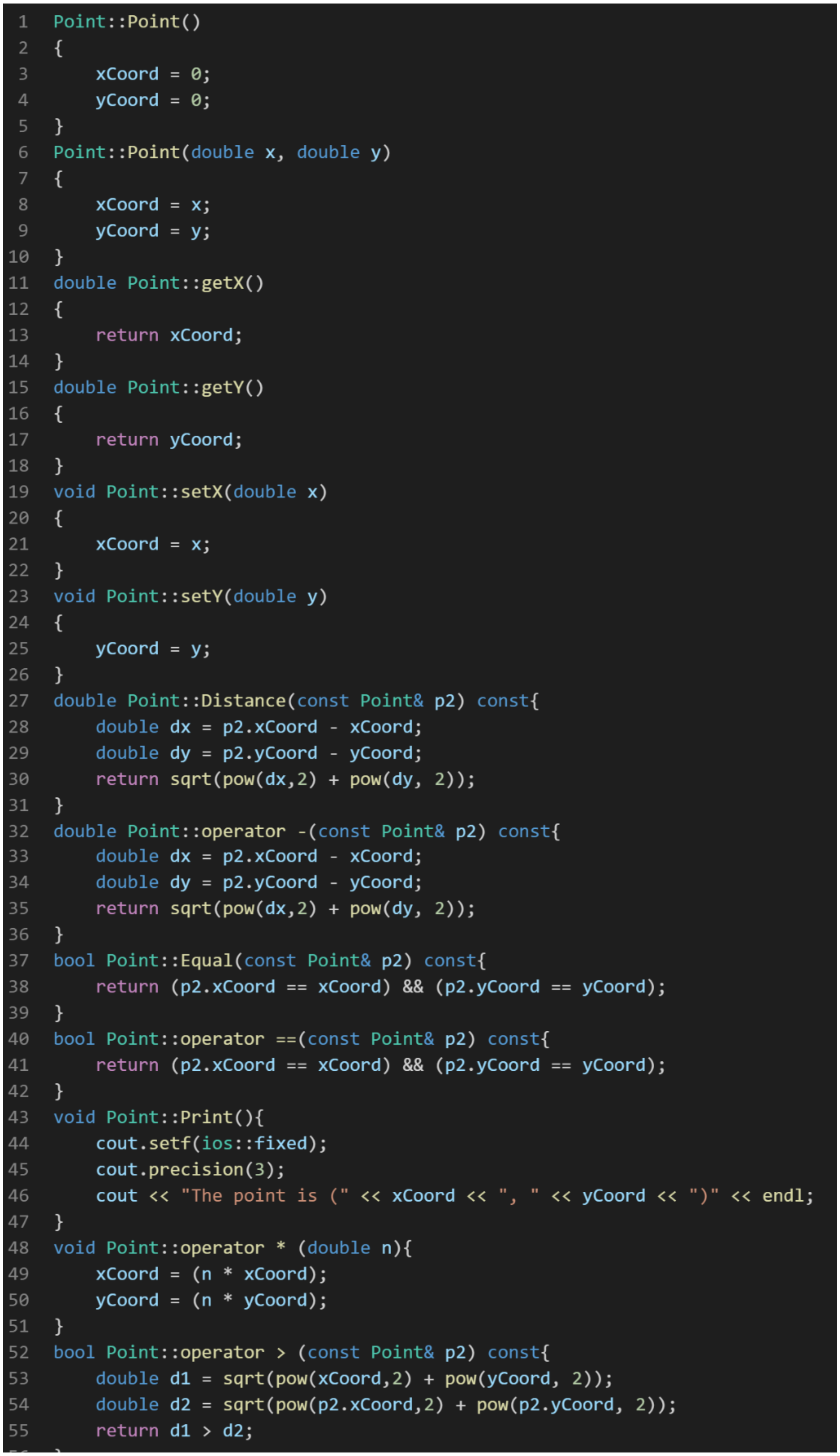
Professor: Xin Wang

Part 1:
main.cpp:



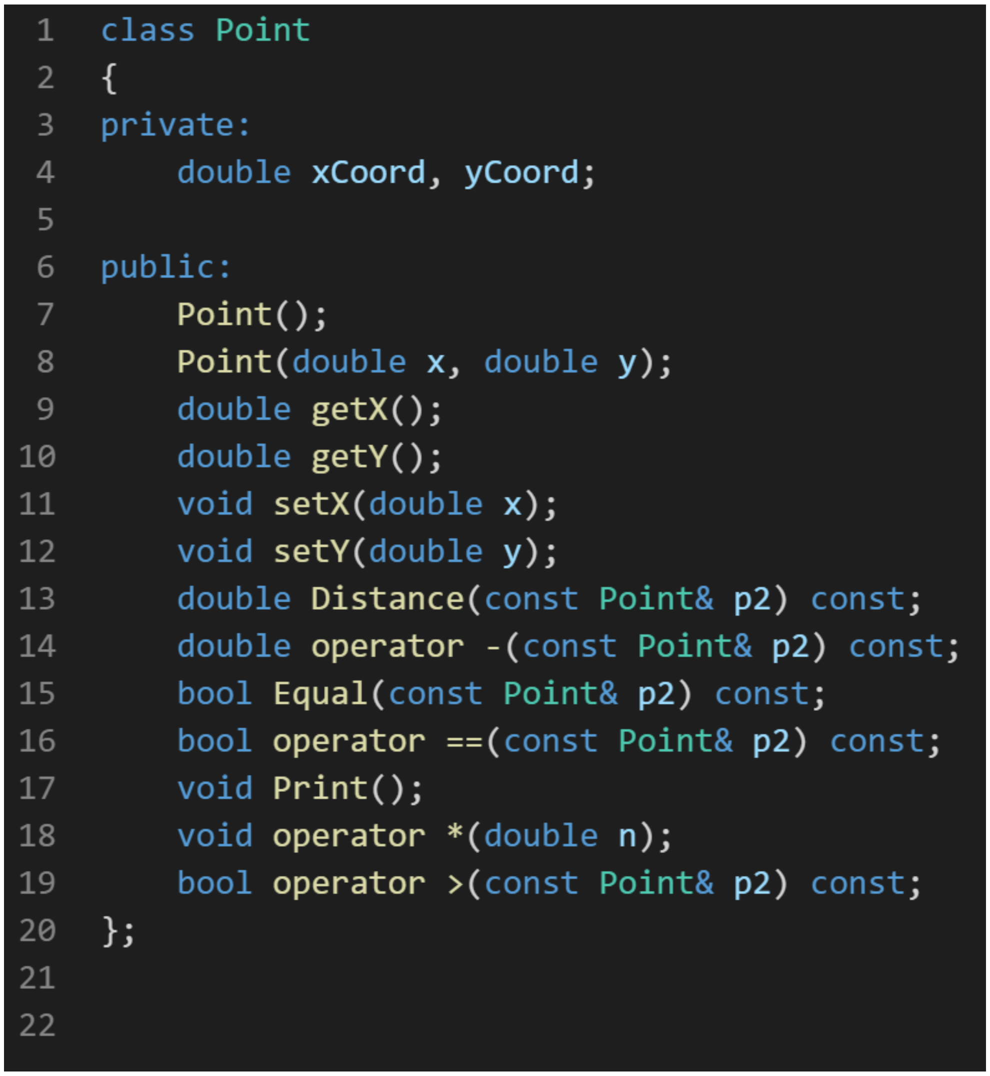
```cpp
double x, y, r;
   cout << "Enter the coordinates of p1: " << endl;
   cin >> x >> y;
   Point p1(x, y);
   p1.Print();
   cout << "Enter the coordinates of p2: " << endl;
   cin >> x >> y;
   Point p2(x, y);
   p2.Print();

   cout << "The distance between the two point is " << p1.Distance(p2) << endl;
   cout << "The distance between the two point is " << p1 - p2 << endl;
   cout << "Are the two point the same? The answer is " << p2.Equal(p1) << endl;
   cout << "Are the two point the same? The answer is " << (p1==p2) << endl;
   cout << "Enter a number to multiplr p2 by: \n";
   cin >> r;
   p2 * r;
   p2.Print();
   cout << "p1 > p2 ? " << (p1 > p2) << endl;
```

Point.cpp



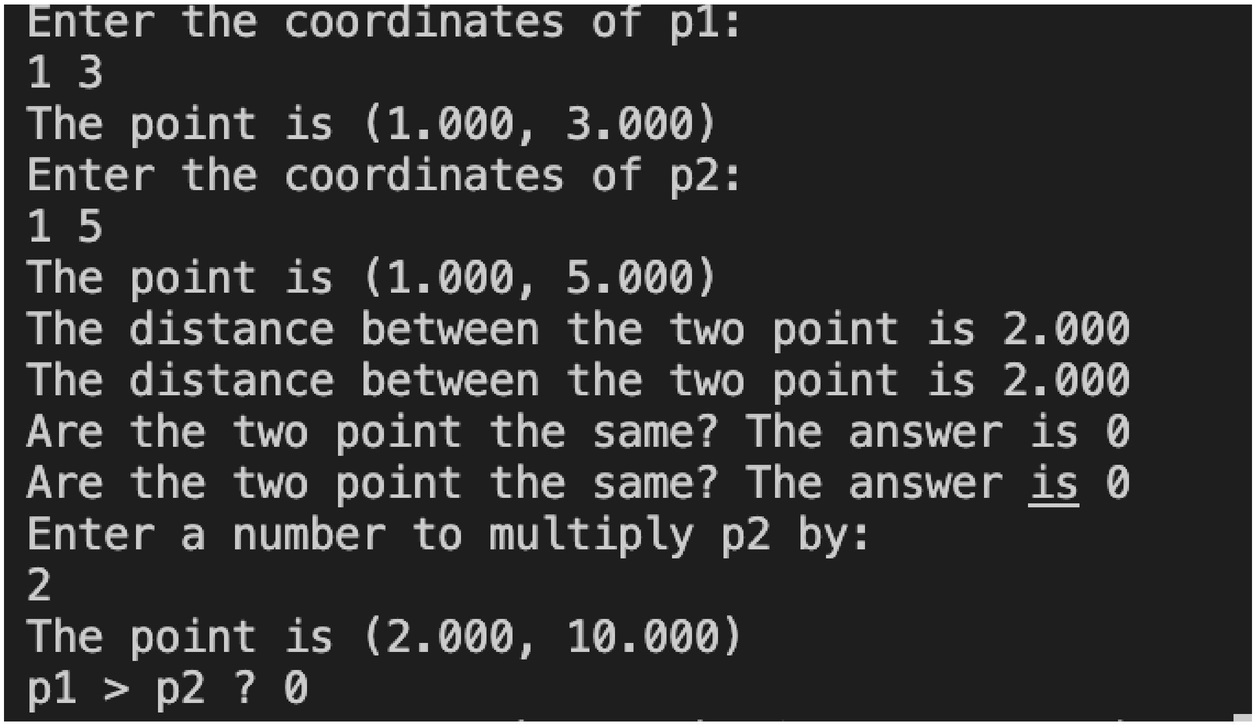
```cpp
Point::Point()
{
    xCoord = 0;
    yCoord = 0;
}
Point::Point(double x, double y)
{
    xCoord = x;
    yCoord = y;
}
double Point::getX()
{
    return xCoord;
}
double Point::getY()
{
    return yCoord;
}
void Point::setX(double x)
{
    xCoord = x;
}
void Point::setY(double y)
{
    yCoord = y;
}
double Point::Distance(const Point& p2) const{
    double dx = p2.xCoord - xCoord;
    double dy = p2.yCoord - yCoord;
    return sqrt(pow(dx,2) + pow(dy, 2));
}
double Point::operator -(const Point& p2) const{
    double dx = p2.xCoord - xCoord;
    double dy = p2.yCoord - yCoord;
    return sqrt(pow(dx,2) + pow(dy, 2));
}
bool Point::Equal(const Point& p2) const{
    return (p2.xCoord == xCoord) && (p2.yCoord == yCoord);
}
bool Point::operator ==(const Point& p2) const{
    return (p2.xCoord == xCoord) && (p2.yCoord == yCoord);
}
void Point::Print(){
    cout.setf(ios::fixed);
    cout.precision(3);
    cout << "The point is (" << xCoord << ", " << yCoord << ")" << endl;
}
void Point::operator * (double n){
    xCoord = (n * xCoord);
    yCoord = (n * yCoord);
}
bool Point::operator > (const Point& p2) const{
    double d1 = sqrt(pow(xCoord,2) + pow(yCoord, 2));
    double d2 = sqrt(pow(p2.xCoord,2) + pow(p2.yCoord, 2));
    return d1 > d2;
```

Point.h



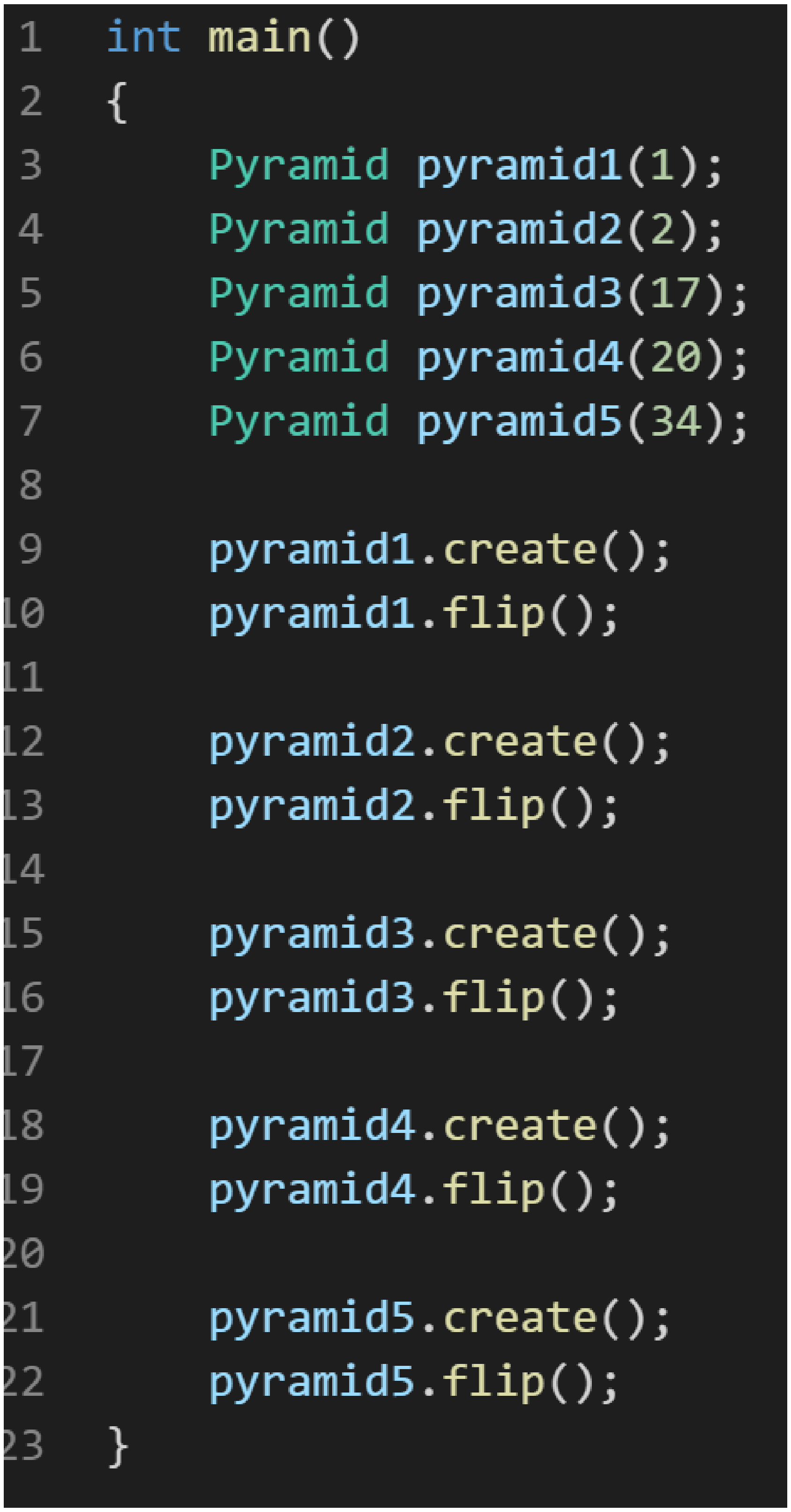
```cpp
class Point
{
private:
    double xCoord, yCoord;

public:
    Point();
    Point(double x, double y);
    double getX();
    double getY();
    void setX(double x);
    void setY(double y);
    double Distance(const Point& p2) const;
    double operator -(const Point& p2) const;
    bool Equal(const Point& p2) const;
    bool operator ==(const Point& p2) const;
    void Print();
    void operator *(double n);
    bool operator >(const Point& p2) const;
};


```

Output:



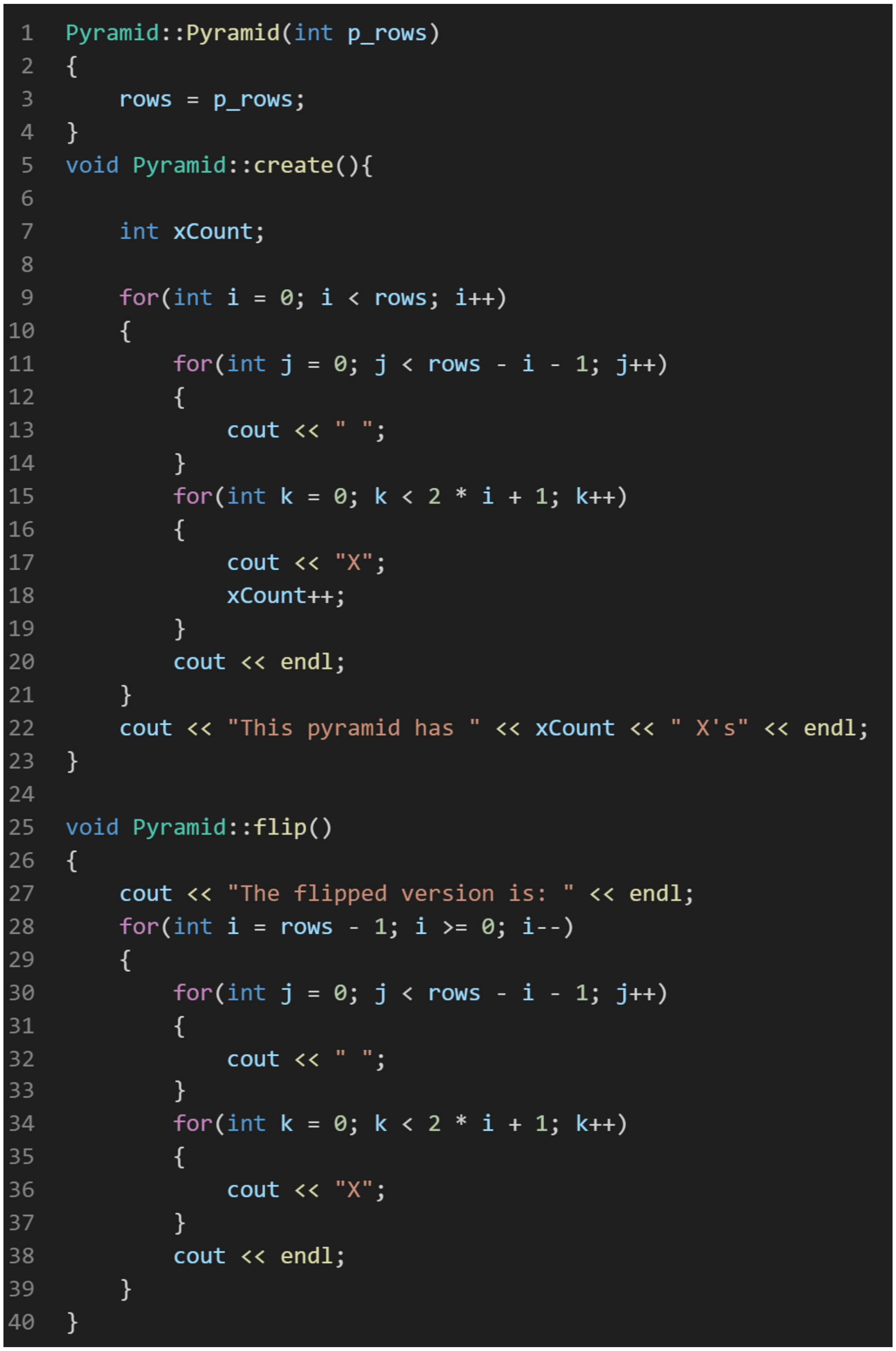
```
Enter the coordinates of p1:
1 3
The point is (1.000, 3.000)
Enter the coordinates of p2:
1 5
The point is (1.000, 5.000)
The distance between the two point is 2.000
The distance between the two point is 2.000
Are the two point the same? The answer is 0
Are the two point the same? The answer is 0
Enter a number to multiply p2 by:
2
The point is (2.000, 10.000)
p1 > p2 ? 0
```

Part 3
Main.cpp



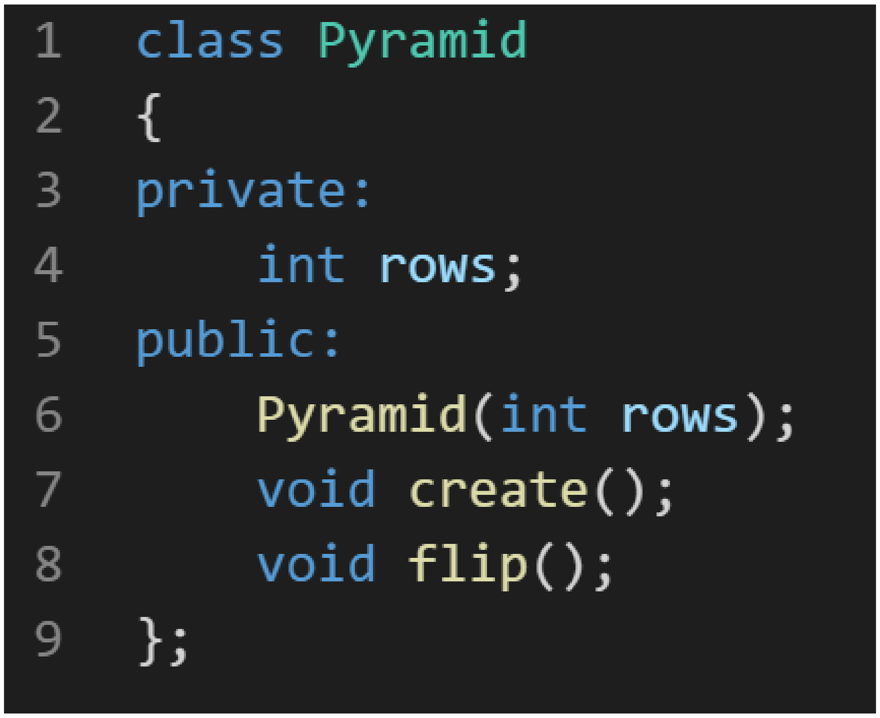
```cpp
int main()
{
    Pyramid pyramid1(1);
    Pyramid pyramid2(2);
    Pyramid pyramid3(17);
    Pyramid pyramid4(20);
    Pyramid pyramid5(34);

    pyramid1.create();
    pyramid1.flip();

    pyramid2.create();
    pyramid2.flip();

    pyramid3.create();
    pyramid3.flip();

    pyramid4.create();
    pyramid4.flip();

    pyramid5.create();
    pyramid5.flip();
}
```

Pyramid.cpp



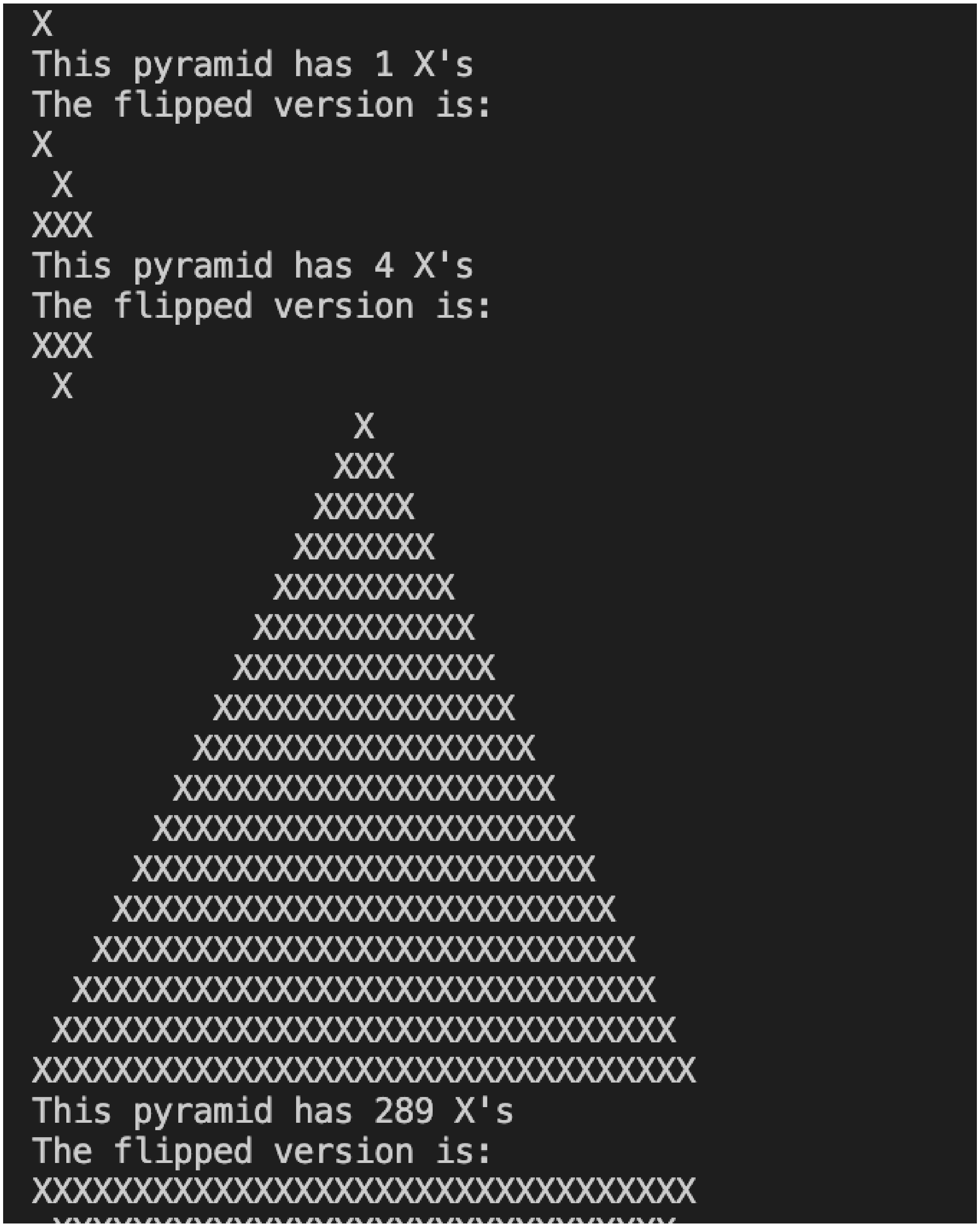
```cpp
Pyramid::Pyramid(int p_rows)
{
    rows = p_rows;
}
void Pyramid::create(){

    int xCount;

    for(int i = 0; i < rows; i++)
    {
        for(int j = 0; j < rows - i - 1; j++)
        {
            cout << " ";
        }
        for(int k = 0; k < 2 * i + 1; k++)
        {
            cout << "X";
            xCount++;
        }
        cout << endl;
    }
    cout << "This pyramid has " << xCount << " X's" << endl;
}

void Pyramid::flip()
{
    cout << "The flipped version is: " << endl;
    for(int i = rows - 1; i >= 0; i--)
    {
        for(int j = 0; j < rows - i - 1; j++)
        {
            cout << " ";
        }
        for(int k = 0; k < 2 * i + 1; k++)
        {
            cout << "X";
        }
        cout << endl;
    }
}
```

Pyramid.h



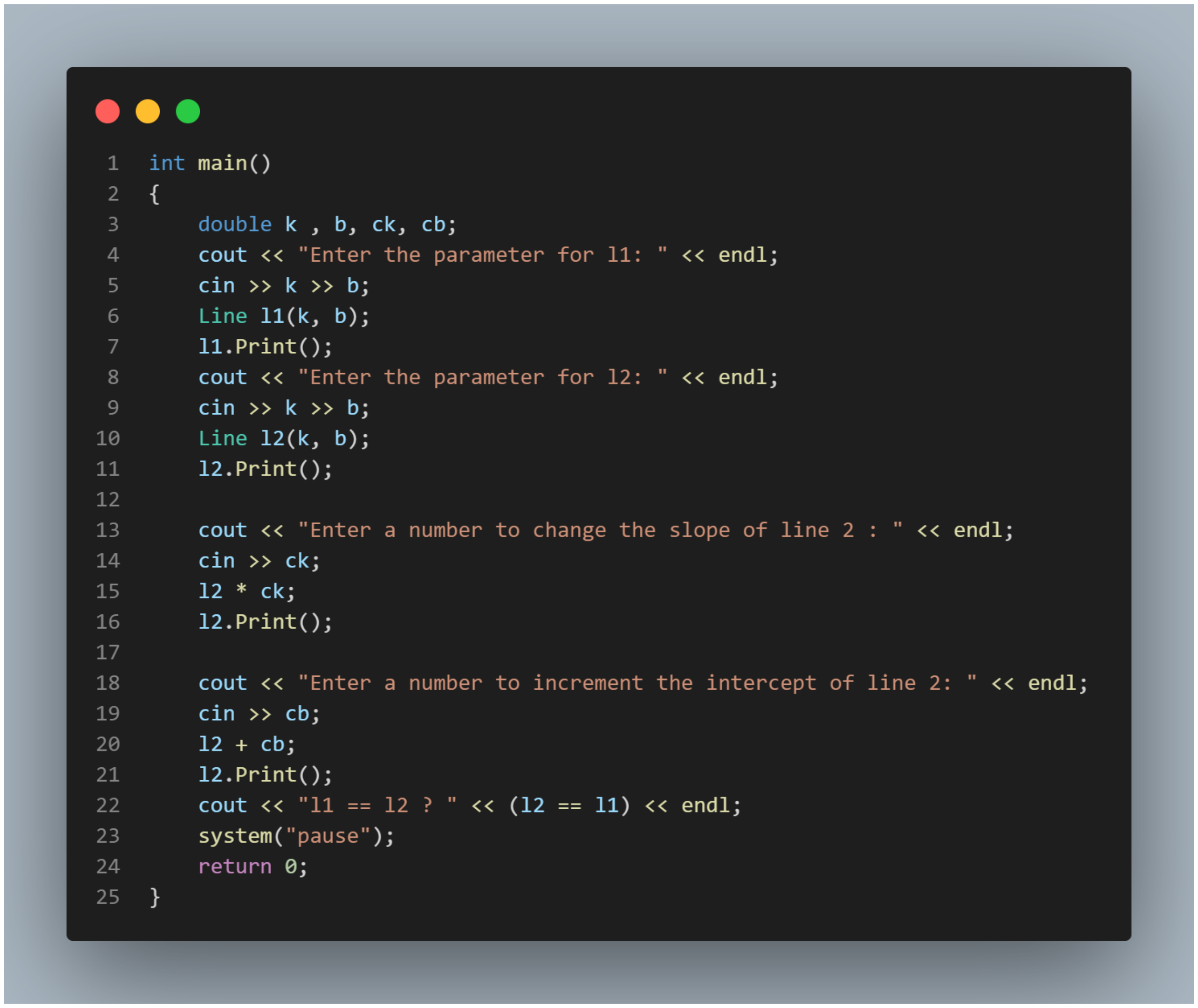
```
class Pyramid
{
private:
	int rows;
public:
	Pyramid(int rows);
	void create();
	void flip();
};
```

Output:



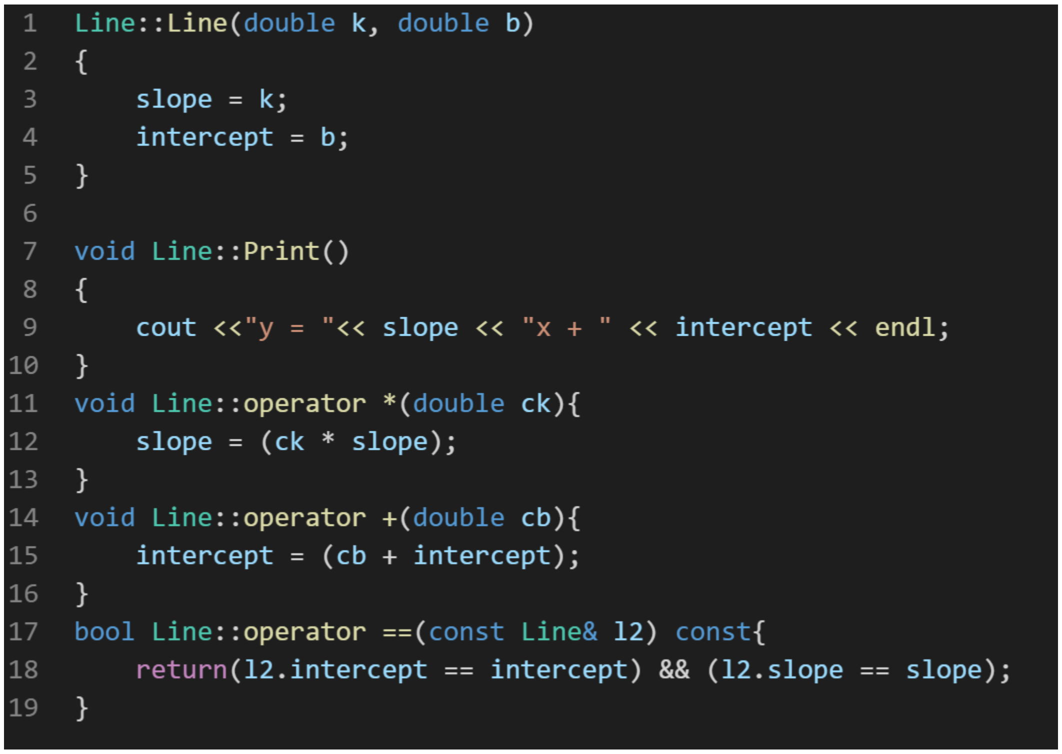
```
X
This pyramid has 1 X's
The flipped version is:
X
 X
XXX
This pyramid has 4 X's
The flipped version is:
XXX
 X
                X
               XXX
              XXXXX
             XXXXXXX
            XXXXXXXXX
           XXXXXXXXXXX
          XXXXXXXXXXXXX
         XXXXXXXXXXXXXXX
        XXXXXXXXXXXXXXXXX
       XXXXXXXXXXXXXXXXXXX
      XXXXXXXXXXXXXXXXXXXXX
     XXXXXXXXXXXXXXXXXXXXXXX
    XXXXXXXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXXXXXXXXXXXXXXX
  XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
This pyramid has 289 X's
The flipped version is:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Problem 4
Main.cpp



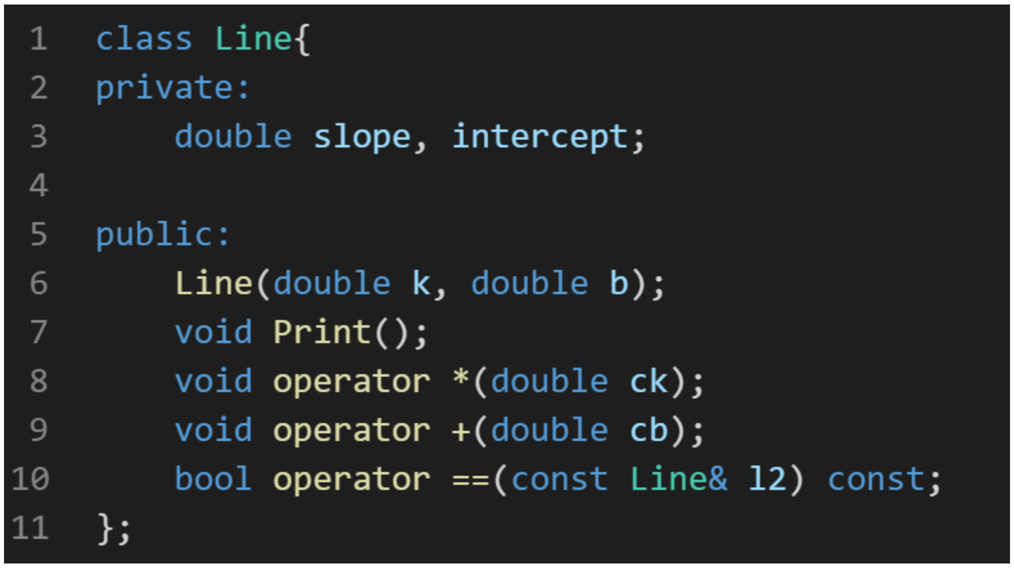
```cpp
int main()
{
    double k , b, ck, cb;
    cout << "Enter the parameter for l1: " << endl;
    cin >> k >> b;
    Line l1(k, b);
    l1.Print();
    cout << "Enter the parameter for l2: " << endl;
    cin >> k >> b;
    Line l2(k, b);
    l2.Print();

    cout << "Enter a number to change the slope of line 2 : " << endl;
    cin >> ck;
    l2 * ck;
    l2.Print();

    cout << "Enter a number to increment the intercept of line 2: " << endl;
    cin >> cb;
    l2 + cb;
    l2.Print();
    cout << "l1 == l2 ? " << (l2 == l1) << endl;
    system("pause");
    return 0;
}
```

Line.cpp



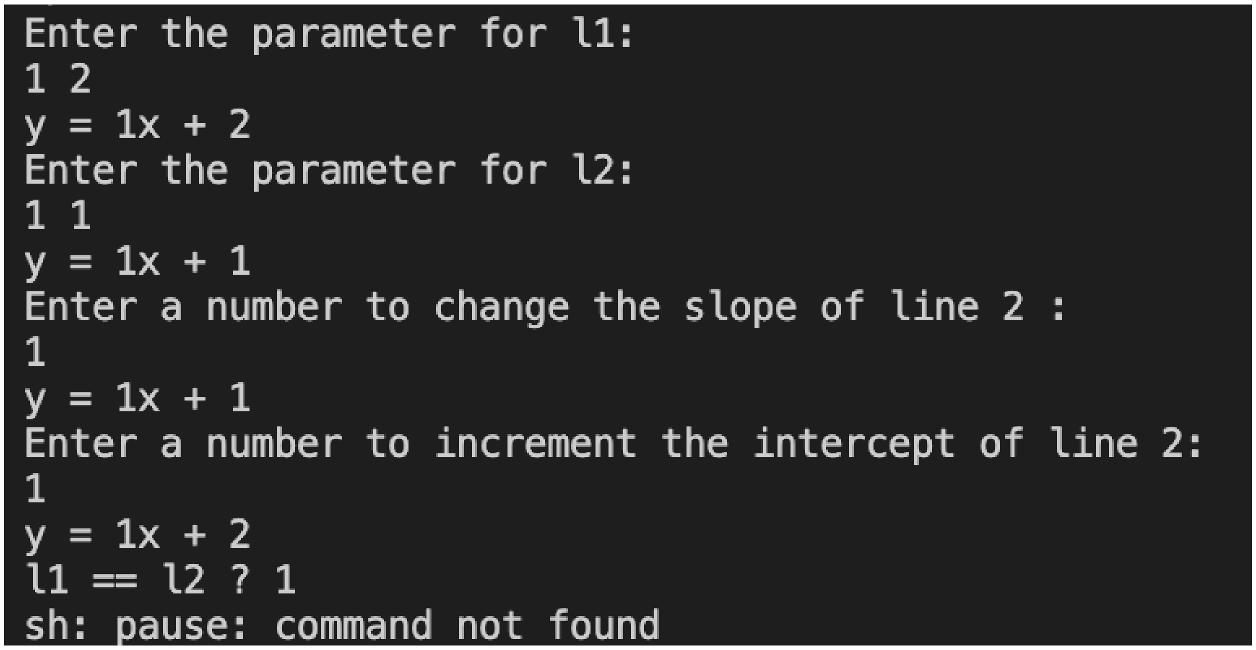
```cpp
Line::Line(double k, double b)
{
    slope = k;
    intercept = b;
}

void Line::Print()
{
    cout <<"y = "<< slope << "x + " << intercept << endl;
}
void Line::operator *(double ck){
    slope = (ck * slope);
}
void Line::operator +(double cb){
    intercept = (cb + intercept);
}
bool Line::operator ==(const Line& l2) const{
    return(l2.intercept == intercept) && (l2.slope == slope);
}
```

Line.h



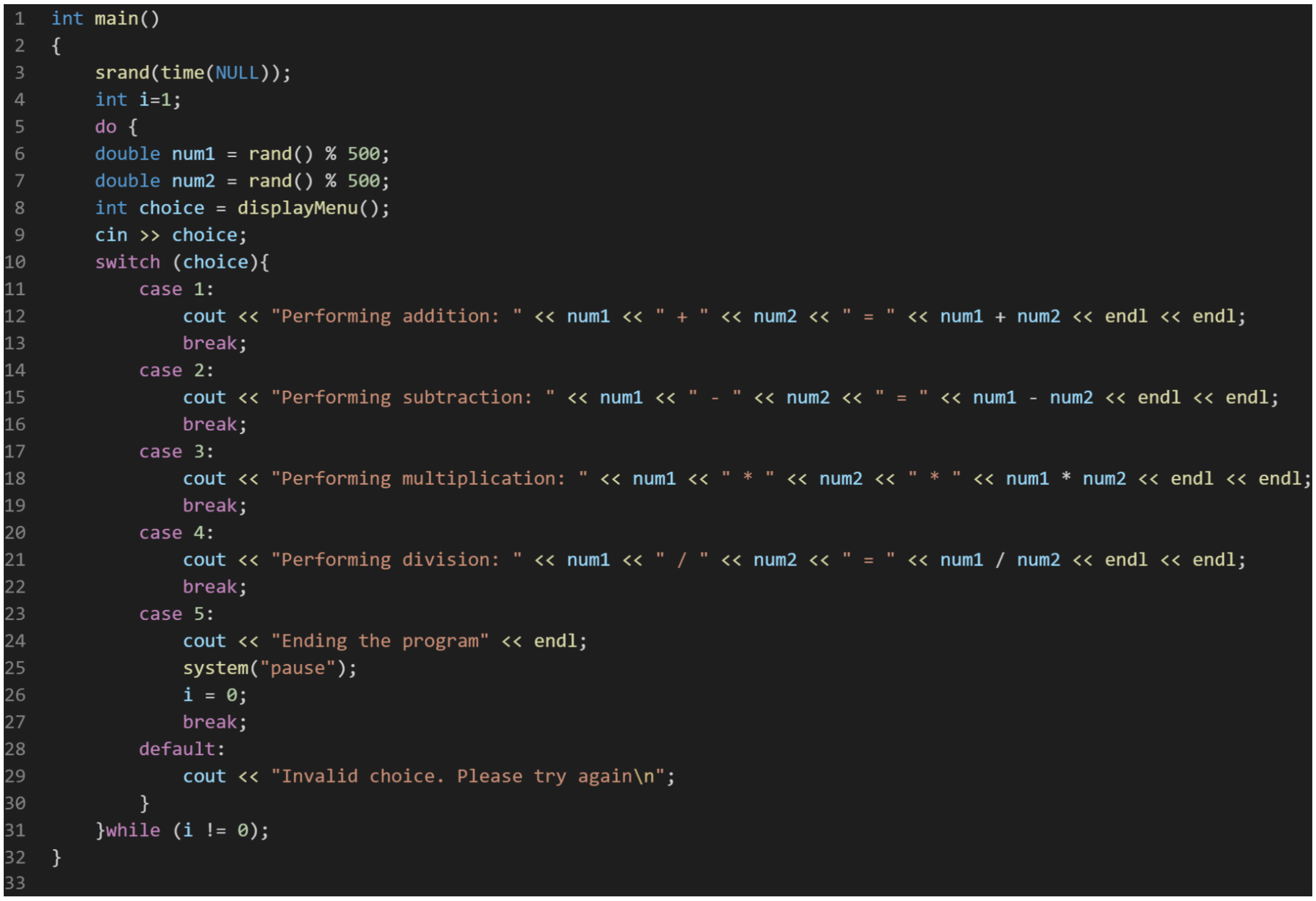
```cpp
class Line{
private:
    double slope, intercept;

public:
    Line(double k, double b);
    void Print();
    void operator *(double ck);
    void operator +(double cb);
    bool operator ==(const Line& l2) const;
};
```

Output:

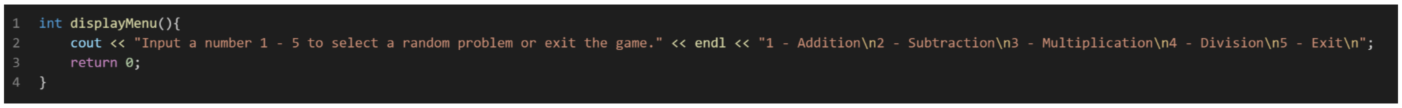


```
Enter the parameter for l1:
1 2
y = 1x + 2
Enter the parameter for l2:
1 1
y = 1x + 1
Enter a number to change the slope of line 2 :
1
y = 1x + 1
Enter a number to increment the intercept of line 2:
1
y = 1x + 2
l1 == l2 ? 1
sh: pause: command not found
```

Part 5:
Main.cpp

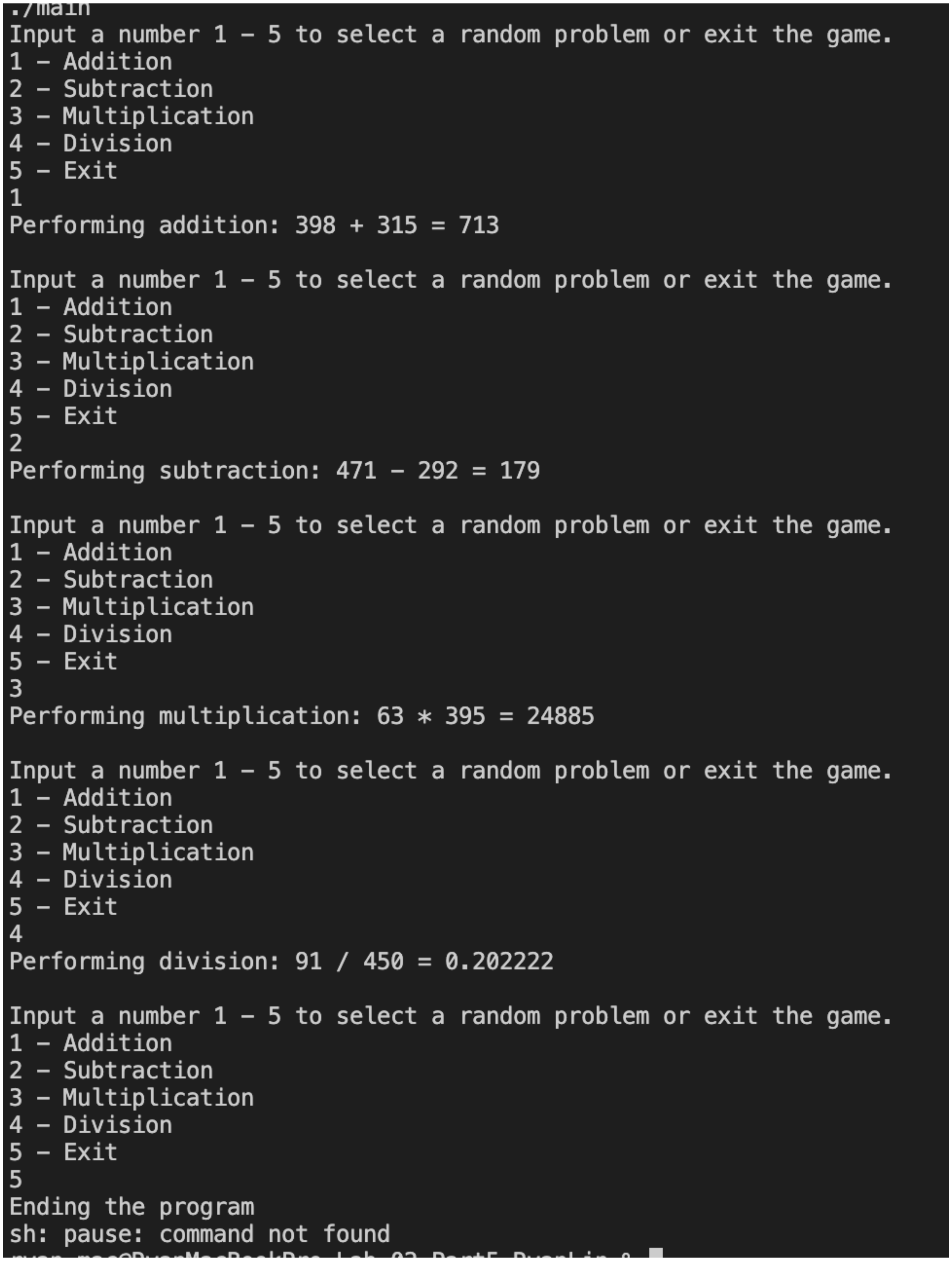


```cpp
int main()
{
    srand(time(NULL));
    int i=1;
    do {
    double num1 = rand() % 500;
    double num2 = rand() % 500;
    int choice = displayMenu();
    cin >> choice;
    switch (choice){
        case 1:
            cout << "Performing addition: " << num1 << " + " << num2 << " = " << num1 + num2 << endl << endl;
            break;
        case 2:
            cout << "Performing subtraction: " << num1 << " - " << num2 << " = " << num1 - num2 << endl << endl;
            break;
        case 3:
            cout << "Performing multiplication: " << num1 << " * " << num2 << " * " << num1 * num2 << endl << endl;
            break;
        case 4:
            cout << "Performing division: " << num1 << " / " << num2 << " = " << num1 / num2 << endl << endl;
            break;
        case 5:
            cout << "Ending the program" << endl;
            system("pause");
            i = 0;
            break;
        default:
            cout << "Invalid choice. Please try again\n";
        }
    }while (i != 0);
}

```



```cpp
int displayMenu(){
    cout << "Input a number 1 - 5 to select a random problem or exit the game." << endl << "1 - Addition\n2 - Subtraction\n3 - Multiplication\n4 - Division\n5 - Exit\n";
    return 0;
}
```

Output:



```
./main
Input a number 1 - 5 to select a random problem or exit the game.
1 - Addition
2 - Subtraction
3 - Multiplication
4 - Division
5 - Exit
1
Performing addition: 398 + 315 = 713

Input a number 1 - 5 to select a random problem or exit the game.
1 - Addition
2 - Subtraction
3 - Multiplication
4 - Division
5 - Exit
2
Performing subtraction: 471 - 292 = 179

Input a number 1 - 5 to select a random problem or exit the game.
1 - Addition
2 - Subtraction
3 - Multiplication
4 - Division
5 - Exit
3
Performing multiplication: 63 * 395 = 24885

Input a number 1 - 5 to select a random problem or exit the game.
1 - Addition
2 - Subtraction
3 - Multiplication
4 - Division
5 - Exit
4
Performing division: 91 / 450 = 0.202222

Input a number 1 - 5 to select a random problem or exit the game.
1 - Addition
2 - Subtraction
3 - Multiplication
4 - Division
5 - Exit
5
Ending the program
sh: pause: command not found
```