
Homework 3

Directions: Write the following 3 programs, and submit your source code to me via Blackboard, using the template files I have provided. You should send only the source code, in cpp format; do NOT send your .exe files. READ THE INSTRUCTIONS on how to submit your work in the Course Documents section of Blackboard.

This programming assignment is meant to give a practice with `cin` and `cout` and `if` statements.

1) linear.cpp

Write a program that has the user input two points in the xy -plane, and then displays the equation of the line through those points in $y = mx + b$ form.

This should involve some very minor algebra, so it should be easy, but there's one catch: I want you to design your program so that the user will enter each point in the following format: (2, 3). That is, an opening (, a number, a comma, a number, a closing).

So, your interface should look like:

Enter the first point: (1.1,2)

Enter the second point: (2.6,8)

The equation is: $y = 4x - 2.4$

(Here, the (1.1,2) and (2.6,8) are values chosen and input by the user; they should NOT be part of your code. The rest of the above is output shown.)

Big Hint: you might want to break this problem down into two parts: the part where you read in the point in the desired format and store the coefficients into variables; and the part where, once you have the coordinates, you calculate the equation. You can try to write a program that does one of these problems first, and then adapt it to solve both problems.

Smaller Hints: For the part where you read in the point, keep in mind that you need to find something to do with all the characters that are input (specifically, the characters from the input that you don't need!). Also, think about data types throughout this program.

Specifications: your program must

- clearly ask the user to enter two points, and then accept any two points in the xy -plane (so long as they have different x -coordinates) from the user, who will enter the points in the form shown – e.g., (3,4).
- clearly display the equation of the line through those two points in $y = mx + b$ form as shown above. “Ugly” output, e.g. $y = 0x + -5$ instead of $y = -5$ is acceptable.

Bonus: try to make your output pretty, by considering all the special cases that might make the output look funny ($y = 0x + -5$ vs $y = -5$).

2) quadratic.cpp

You can pretty much find the solution to this problem in at least one of the textbooks I suggested at the beginning of the term. However, I **strongly** recommend that you do this exercise yourself before consulting the text or anyone else.

Write a program that prompts the user to input the three coefficients of a quadratic equation, and then displays all the solutions of the quadratic with *4 digits printed out after the decimal place*. If there is one solution, then it shouldn't be printed twice. If there are only complex solutions, then these solutions should be displayed in the form $a + bi$: your answer should be displayed like, e.g., $x = 4.1523 + 2.7610i$. If the equation I have entered is not a “true” quadratic (because the x^2 coefficient is zero), the program should tell me that I did something wrong.

For example: suppose I wanted to solve $3x^2 + 5x + 1 = 0$. Your program should prompt me to enter the coefficients, and I would enter 3, 5, and 1. It should then display the two solutions: in this case, the program should display $x = -0.2324$ and $x = -1.4343$. If I wanted to solve $x^2 + 2x + 1 = 0$, I would enter 1, 2 and 1; and $x = -1.0000$ should be displayed (once). If I wanted to solve $2.1x^2 + 4x + 10 = 0$, I would enter 2.1, 4, and 10, and $x = -0.9524 - 1.9634i$ and $x = -0.9524 + 1.9634i$ should be displayed.

Specifications: your program must

- ask for and accept the coefficients from the user, in a *clear* manner: when I run the program, I want to be able to tell what I am supposed to do from what is displayed on the console.
- display two solutions with 4 digits printed after the decimal if there are two real solutions to the quadratic with given coefficients (and of course, there should be no *i* displayed in this case).
- display one solution, once, with 4 digits printed after the decimal if there is only one real solution. (For this one, it is ok if it doesn't work all the time: why?)
- if the two solutions of the quadratic are not real, display the solution in the form $x = a + bi$, where a and b are each displayed with 4 decimal places.
- if the x^2 coefficient given is 0, print a message saying so.

Hint: think about the order in which you should address the four different display specifications in your program.

3) parks.cpp

The “center” of Central Park is located at latitude 40.782800 degrees, longitude -73.965269 degrees.

The “center” of Prospect Park is located at latitude 40.660217 degrees, longitude -73.968948 degrees.

The “center” of Flushing Meadows Corona park is located at latitude 40.739694 degrees, longitude -73.840793 degrees.

(By “center” I mean “points in the middle of the park that I randomly chose.”)

Write a program that asks for a latitude and longitude from the user, and prints out which park is the closest to that point, by computing the distances from that point to the three given ones. Don't worry about the very unlikely possibility of a tie.

We assume that the point entered is in or close to New York City, so that we can treat the Earth as essentially flat. For those who struggled with the problem on Homework Assignment 1, here is the scheme for calculating distances under these assumptions, expressed in terms of kilometers instead of miles. To calculate the distance between two points:

1. Calculate Δ_{Lat} , the difference between the latitudes. Multiply by 111.048 to convert the degree difference to kilometers.
2. Calculate Δ_{Long} , the difference between the longitudes. Multiply by 84.515 to convert the degree difference to kilometers.
3. The distance between the points is $\sqrt{(\Delta_{Lat})^2 + (\Delta_{Long})^2}$.

The strategy is simple: calculate three distances, and then use some **if** statements – or **if-else**, or nested **if**, whatever you find to be most appropriate – to determine which distance is smallest, and print out that park.

Your interface should look like:

Enter Latitude: 40.743

Enter Longitude: -73.999

Central Park is the closest park.

Here, 40.743 and -73.999 are entered by the user; everything else is printed out by the program. Be sure to test your program with other points, too!

Specifications: your program must

- ask for and accept the latitude and longitude of a point within New York City.
- display the name of the park corresponding to the closest location, as calculated by the scheme described above.