ECE120: Introduction to Computing   Optional C Exercise for Week 3

Implementing a Multi-Region Function

Overview:

In this exercise, you will implement an integer function F(x,y,...) on 2, 3, or 4 integer variables. The function F is different for each student, so please check the specifics of your unique function in “mp_spec.txt”, a file in your working directory. To check out the files of this exercise, please use the “svn update” command in your ECE120 working directory, and all the files for this exercise will appear in the folder named “OPTIONAL3” in your working directory. Your function should consider the first two input integers (x and y), as a pair of coordinates. The number of integer variables that you should take as inputs is also specified in “mp_spec.txt”.

Specification:

You should implement the C function main in the file “practice.c” to do the following:

1. Use printf to print a prompt “Please input N integers.\n”. The N is the number of input integers. For example, print “Please input 2 integers.\n” if your “mp_spec.txt” file states that you should take 2 integers as your input variables.
2. Use scanf to read input values for the parameters x, y, j, and k in order. Use fewer parameters if appropriate for your individual function. For example, read in x, y, and j, if your number of input integers is 3.
3. Calculate the output value of function F(x,y, …), —see below for details.
4. Print the output of function F using a single printf: "F(%d,%d, …) = %d\n”. For example, please print out “F(-10,1) = 368\n”, if your input values are x = -10 and y = 1 and the result of your function F is 368. (Please notice that there is no space between the numbers and the commas, but there are spaces around = .)

Function F:

The following constraints divide a two dimensional plane into eight regions (see Figure 1). The value of function F depends on region, and uses two of these constraints to determine in which region a point (x, y) lies:

- A circle C whose center is at the origin (described by $x^2 + y^2 = r^2$).
- A linear constraint (line G) passing through the origin (0, 0).
- A linear constraint (line H) not passing through (0, 0), but crossing through the circle C so that it intercepts the first linear constraint (line G) inside of the circle C.

Illustration of Constraints

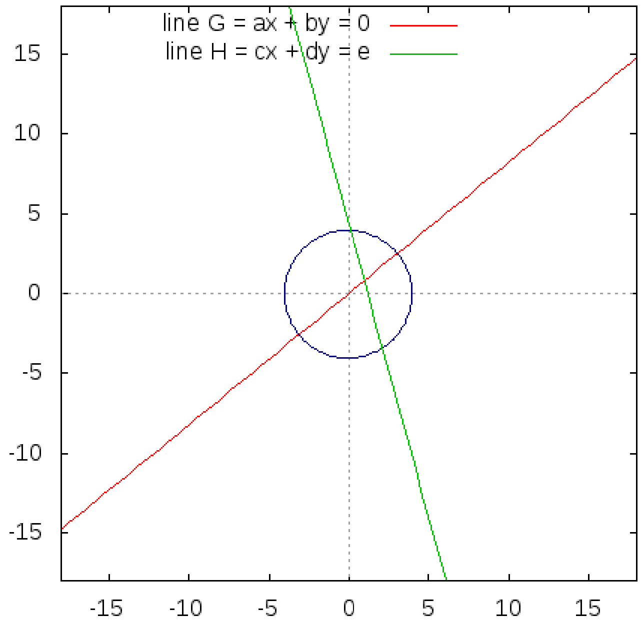

Figure 1: The coefficients of all the constraints for your individual functions are defined in “mp_spec.txt”. They are not the same as the values illustrated here.

The output value of function F depends on the region in which the point (x,y) lies, and the value is calculated by a mathematical function. Types of outputs consist of the following:

1. Linear polynomials of several variables (for example, -68x+ 3y - (-74j)).
2. Sum over values from 1 to the absolute value of <the last parameter> of i^3 * <one other parameter>. For example: $\sum_{i=1}^{y} i^3 * x$. (Note, you can use the abs() function in the standard C library to get the absolute value of an integer. For example, abs(-10) returns 10. And the i^3 is just i*i*i.)

Hint:

Both lines G and H can be written as: ax + by + c = 0, where a, b, and c are integers (unrelated to those variables in Figure 1). Suppose (x_0, y_0) is an arbitrary point A on one of the lines. Then the value of $ax_0 + by_0 + c$ (with a, b, and c for the line on which A lies) must be 0. Let B = (x_0, y_1) be a point directly above or below point A in the plane. Note that B has the same X coordinate as point A. If we now calculate the polynomial for point B, we have $ax_0 + by_1 + c = b(y_1 – y_0)$. If b is positive, the product $b(y_1 – y_0)$ has the same sign as $(y_1 – y_0)$, so a positive value of the equation means that B is above A, and a negative value means that B is below A. If b is negative, these conclusions are reversed. In particular, the product $b(y_1 – y_0)$ has the opposite sign as $(y_1 – y_0)$, so a positive value of the equation means that B is below A, which a negative value means that B is above A. These observations enable one to determine the position of any point on the plane (whether the point is above or below the line) by simply plugging the coordinate of the point into the equation ax + by + c and checking the sign of the result.

Compilation:

You can use the command “gcc -Wall -g practice.c -o opt3” to compile your code.
And use “./opt3” to run your code.
Please note that if the code that you submit cannot be compiled, you will receive no feedback.

Checkout and feedback:

Checkout:

Please use “svn update” in your ECE120 working directory to get the folder “OPTIONAL3” which has all the files. If you do not have a working copy, you can check one out using “svn checkout https://subversion.ews.illinois.edu/svn/fa16-ece120/netid ece120”

Commit:

After you commit your code to your svn repository (using the “svn commit” command), our tool will begin to grade your code. The grading should take a couple of minutes.

Feedback:

Please use the "svn update" command to get the feedback after the grading is completed. The feedback will be in separate files named “*.test”, for example, “0.test”. The feedback files are usually one or two sentences indicating your error. Test cases are also available for some errors. For this exercise, we will first test your printf formats, so please make sure that your printfs follow the requirements. For any questions regarding the test cases and this exercise, please email me at beipang2@illinois.edu or to Prof. Lumetta at lumetta@illinois.edu.

If the feedback has only one file “0.test” which says you have passed all the tests, then congratulations, you have completed this exercise.