EECS 211 – Winter Quarter, 2014
Lab 1

Due: 11:59 pm, Tuesday, January 21, 2014

In this assignment, you will develop a program for generating increasing arithmetic sequences. Your code will be able to output multiple sequences desired by the user, as well as checking for errors in the user's input.

You program should have the following features:

- Each arithmetic sequence will be defined by a minimum value, a maximum value, and either a step size or number of elements in the sequence.
- The minimum and maximum will be entered first, in that order.
- If the min value is larger than the max, your program should output the error message "Invalid max and min values" on a new line.
- Sequences defined by a step size (delta) will follow the min and max by the letter 'd' followed by the step size.
    - If the step size is greater than 0, your program should output the minimum value, then increase this value by the step size and print it until this value is larger than the maximum. Note that the step size might not evenly divide the interval between min and max. Do not print out a value larger than the max.
    - The step size must be an integer.
    - A step size of 0 or less is invalid, and in this case your program should output the message "Invalid value of delta" on a new line.
- Sequences defined by a number of elements will follow the min and max by the letter 'n' followed by the number of elements.
    - If the number of elements is greater than 0, your program should output the minimum value, (number of elements – 2) intermediate values, then the maximum value, where all of the intermediate values are evenly spaced between the max and the min.
    - The number of elements must be an integer.
    - Note that the intermediate values in the sequences might not be integers. Think carefully about what is the interval between these values and how you would generate them.
    - A number of elements of 0 or less is invalid, and a one-element sequence is invalid unless the min and the max are equal. Your program should output the message "Invalid number of elements" on a new line in this case.
- If the third argument for the sequence is not 'd' or 'n', your program should output the error message "Invalid sequence type" on a new line.
- Your program should not print any prompts for input.
- Your program should continue to read sequences from the user until a parsing error occurs (e.g., the word "stop" is encountered rather than a syntactically correct sequence).

- Your program should read all input from the file p1input.txt. Code for reading values from this file will be provided to you.

A test data file (p1input.txt) and its correct output (p1output.txt) are attached to this document. You should examine these to make sure that you understand the format of the data and what to expect as output when your program is correct.

Use the attached main.cpp file as the basis for your work. Add your code, declarations, and comments to this file. NOTE: When the TAs grade your program or any of us help answer your questions, we will simply copy your main.cpp into a project that we have already created, rebuild the project, and run. If your files are not named exactly as indicated – main.cpp and p1input.txt – this process will not work, and we will not be able to respond to your questions or grade your work.

Submit your main.cpp file to the Lab 1 drop box *corresponding to your IDE* (Visual Studio or Xcode) by midnight of the due date. Submit only the main.cpp file. Do not include any of the intermediate files generated by your IDE.

Comments and Suggestions:

We have provided a function (getValidSequence) that can be used when you are developing your solution. When developing and testing software it is sometimes useful to be able to use the keyboard rather than have data come from a file. This allows the developer to rapidly test different scenarios without having to constantly edit a text file. In the case of this assignment, changing the last argument of getValidSequence (file) to cin will accept input from the keyboard rather than p1input.txt.

It is rarely a good idea to try to develop an entire program in one step. It is much better to identify small steps that you can make in succession and do them one at a time. You implement the first small step and test it and fix any mistakes. When that much seems ok, add in the second step, and so on. This approach is compatible with the top-down development method described in a separate document. In this assignment the top-down development approach breaks the assignment into several independent steps that are each quite easy.

1. Step 1: the main loop. The main loop proceeds until an invalid sequence is encountered or there are no more sequences to process (end of file). Write an outer loop whose body simply prints a message describing the sequence, without actually calculating it. Test this much to see if you can correctly tell when to quit.

2. The body of the main loop processes one sequence. This involves reading in the sequence arguments, then calculating the sequence values based on the value of delta or the number of elements. You could implement this in stages:
   a. First, add code to test that the min and max are valid. Test that these errors are being detected correctly.

b. Then, just test the sequence type to determine if delta or the number of elements is being provided. For this step, you could simply print out a message indicating the sequence type. Add code to output the appropriate error message if the user specifies an invalid sequence type.
c. Next, add code to actually compute the sequence values for the case where the user enters a value for delta. Test to ensure that this code works correctly for cases where delta evenly divides the interval between max and min and where it does not. Make sure to test for an invalid value of delta, as well.
d. Finally, add code to compute sequence values for the case where the user enters a number of elements in the sequence. Test to ensure that this code prints a sequence with the correct number of elements. Make sure to test for an invalid number of elements. You may be able to use a common code to generate and output the sequences in both cases, but this is not necessary
e. .