ECE 2220: System Programming Concepts **Fall 2016**
Machine Problem 1 **Due: 11:59 pm, Thursday, August 25**

The goal of this machine problem is to review C programming. This assignment provides a simple example of prompting for and processing text input from the terminal, manipulation of arrays, and utilization of conditionals and loops. The interactive approach to prompting the user for data, reading input from the terminal and printing to the terminal will be utilized in many of the programming assignments this semester. It is critical to adhere precisely to the format of the input and output because we will utilize automatic tools to test for proper operation of your code. A template is provided that contains some of the formatted print statements. You must use the template and these **`printf`** statements. Do not change their text, though you can add additional print statements.

To receive a transmission, a radio must detect the start of the incoming waveform. In wireless networks there may often be multiple overlapping transmissions, and under certain circumstances a receiver must be able to pick out one of the waveforms as its desired transmission. In these types of systems, each transmission contains a preamble with a unique pseudo-random sequence. One of the first tasks of a receiver is to correlate the current incoming waveform against the sequence used by the desired transmission and determine if there is a match. Due to noise in the system and imprecise alignment of the receiver's clock with the incoming waveform, the received data contains errors. Thus, a detection and verification process must be performed to ensure that the receiver is aligned to the start of its desired transmission. In this machine problem we will model a portion of this detection process. We will assume that when a radio is ready to receive, a correlation process is continually looking for the desired transmission. For each estimate of the starting position of the waveform, the correlation process will produce a number indicating the degree of correlation at the current position. The waveform is designed so that there should be multiple matches, and the location of the first match is the start of the desired waveform. However, due to unknowns such as the received power level, we don't know the exact value of a match. Furthermore, due to the presence of other undesired waveforms, there may be a few false alarms.

Your program begins with a configuration step in which the user is prompted to specify two values. The values are read in one time, and are fixed for the remaining duration of the program. First prompt for the correlation threshold. The *correlation-threshold* is an integer and its value must be equal to or larger than the constant `MINTHRESH`. If the input value is invalid, print a message and repeat the request for the correlation threshold until a valid value is entered. However, if the value is -1, then exit your program instead. Next, prompt for the minimum correlation value. The *min-correlation-value* must be a positive integer (greater than zero), and again prompt until a valid value or a -1 is input.

Once the configuration step is over, then the program moves to a sample collection phase in which the user either enters a new set of samples or a -1 to quit the program. If the <u>first</u> sample value is -1, print a message and exit the program without reading further samples. A sample value can have any value (positive or negative), except the first value (which cannot be -1). Loop to read in each sample value one line at a time until you have read in `MAXSAMPLES` or until you have detected `STOPCOUNT` consecutive sample values that are zero. Notice that you do not stop reading sample values if the total number of zero values is `STOPCOUNT` (or more), ONLY if you detect `STOPCOUNT` zeros in a row. After collecting all the samples in a set, perform the detection process (defined below). Once the detection process is over, prompt the user for the next set of samples. Again, exit your program if the first sample value is a -1. Notice that you do <u>not</u> prompt for a new value for *correlation-threshold* or *min-correlation-value* as these configuration values are entered one time only. When reading a set of samples, each value must be entered one line at a time with an end-of-line character between each value (i.e., type enter after each sample value). See the provided input file for an example of valid input.

The detection process is as follows. Detection is successful if there is a sample value that is found *correlation-threshold* or more times, and the value is greater than or equal to *min-correlation-value*. There may be more than one sample value that satisfies this test, and if so select the largest sample value that passes the detection test (note: it is not the value with the most number of matches). If detection occurs, print the value of the sample value, the index position of where the first sample value is found in the set, and the count of the number of times this sample value appears in the set. Assume the set is numbered starting at 1. If there are no sample values that pass the detection test, then print a message that a desired waveform was not detected. For the format of the print statements, see the template file.

Notes

1. Use #defines for the constants. Never hard code parameters into your program. When you submit your code the following definitions must be set. During testing you can experiment with smaller values.

```
#define MAXSAMPLES 500
#define STOPCOUNT  3
#define MINTHRESH  3
```

2. Here is an example of the input a user might type (the format is critical), and the output that must be printed. The user types the numbers in the left column. Of course, your program will print additional output information so that the user is prompted for input values. Your prompts should be designed to make it easy for a user to input the values. We will test with extensive combinations of inputs that will try to break your program.

Sample Input	Minimum Required Output for the Sample Input
3 5	
10 5 2 5 50 5 0 0 0	Waveform detected at position 2 with value 5 and appears 3 times
10 5 5 10 5 10 0 0 0	Waveform detected at position 1 with value 10 and appears 3 times
1000 5 1000 5 5 5 0 0 0	Waveform detected at position 2 with value 5 and appears 4 times
4 0 -3 0 4	

0 4 -3 -1 -3 0 0 0	No waveform detected
-1	Goodbye

3. You compile your code using:

```
gcc -Wall -g lab1.c -o lab1
```

The code you submit must compile using the `-Wall` flag and **no** compiler errors or warnings should be printed. To receive credit for this assignment your code must compile and at a minimum evaluate the set of samples given above. Code that does not compile or crashes before evaluating the first set will not be accepted or graded. Code that correctly detects the examples above but otherwise fails many other cases will be scored at 50% or less. OS X users must do a final check on one of the CES Ubuntu machines to verify that gcc using Ubuntu shows no warning messages.

While your program is designed to be interactive, to verify that your program can read input correctly download the example input file from blackboard and run your program as follows:

```
./lab1 < test_input
```

The output may not be as nice as when you run interactively, but the output must contain the strings shown in the example above. (We will search for these specific lines. Note there are four sets in this example.)

4. Submit your file `lab1.c` to the ECE assign server. You submit by email to ece_assign@clemson.edu. Use as subject header ECE222-1,#1. The 222-1 identifies you as a member of Section 1 (the only section this semester). The #1 identifies this as the first assignment. When you submit to the assign server, verify that you get an automatically generated confirmation email within a few minutes. If you don't get a confirmation email, your submission was not successful. You must include your file as an attachment. You can make more than one submission but we will only grade the final submission.

5. Turn in a paper copy of all code files at the start of the first class meeting following your submission. Print in landscape mode, two columns, with a small monospaced font size to save paper.

See the ECE 222 Programming Guide for additional requirements that apply to all programming assignments.

Work must be completed by each individual student. See the course syllabus for additional policies.