LAB #5

Purpose:

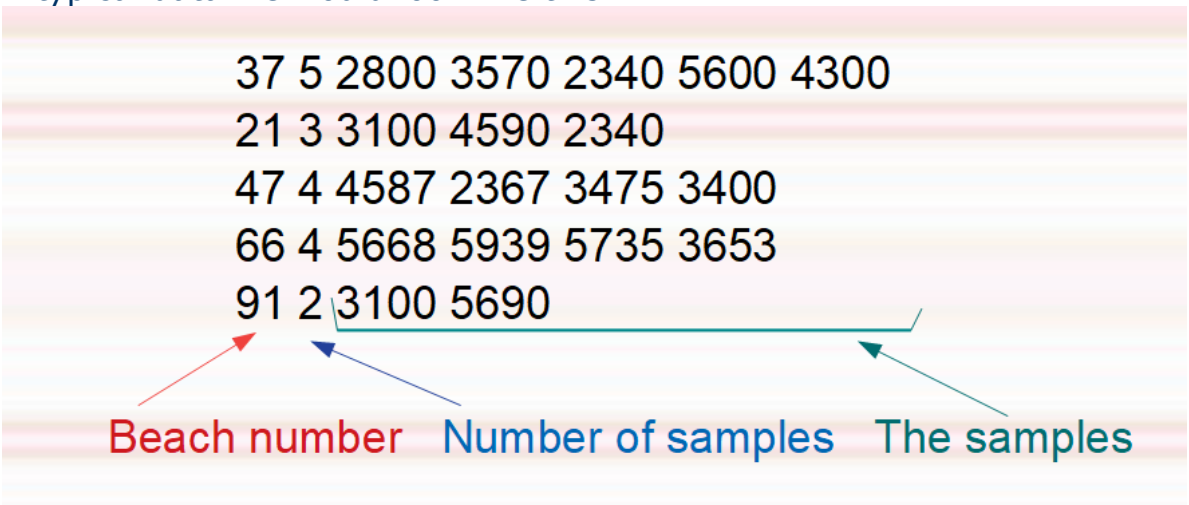
This lab helps you to apply the concepts of iteration and, in particular, nested loops in C, and to provide further practice in implementing selections in C. In this lab you are going to determine, from water quality measurements, whether or not a beach should be closed for swimming. A real world set of exercises.

Before the lab:

1. Read your notes concerning loops with particular attention to while and for loops.
2. Read your notes concerning input data from files (FILE, fopen ,fscanf, flose) or file redirection.

During the lab:

PART I: PROGRAMMING EXERCISES
Exercise #1:
The program we will eventually have written in the lab will calculate the average level of bacteria in the water at various beaches and print out a message if the beach is safe or should be closed for swimming. This type of program is often found on midterm tests and/or the final examination. A similar problem (but not identical) can be found on the example programs website (see programs 7, 8, and 9). We will do this program in 11 steps:
A typical data file would look like this:



a. Write a scanf statement to read in one number (int), and another statement to print out that number.
b. Write a for loop to read in a series of 5 numbers (int), each iteration reading one number and printing it out.

c. Write one scanf to read in three integers into variables standing for beach number (int), number of samples (int), number of organisms per 100 ml of water (int). Choose appropriate names for the variables like: b_num, num_samples, num_orgs_per_100.

d. Write a scanf line to read in an integer "number of samples" from the keyboard into the variable num_samples. Following that, write a for loop to iterate (repeat) num_samples times. In each iteration have your code read in one value from the keyboard into a variable (any name will do, here) and print it out. eg: if num_samples is 8, your code will read a value in the first line of the body of the for loop and then print it out in the next line, repeated a total of 8 times. Note that each new read overwrites the previous value in your variable. REMEMBER: a loop (for or while, and also if and else for that matter) ALWAYS executes ONLY a SINGLE statement - which means that you do NOT need curly braces { } around that one statement. If you want your for loop to execute two statements (scanf in the value, and then print it out), they both must be put into a "compound statement" (any number of statements inside curly braces). This compound statement acts like a single statement insofar as the for loop is concerned, so the single compound statement (that is, the two statements it contains) is executed once each time the for loop iterates.

e. Now slightly modify the for loop code as follows. Continue to use the scanf line to read in the number of samples into the variable num_samples, before the for loop code. However, this time have each read place each value into the variable named num_orgs_per_100 ... and then, as before, print out each in the second line of the compound statement. (Note that each subsequent read will overwrite the value in num_orgs_per_100 from the previous read, just as happened in the previous section. We will do something different with those values, below.)

f. This step introduces and arranges data. Your coding is not changed in this part. In the above, your data has come from the keyboard. You can continue to do that, but putting the data in a file (input with redirection or FILE I/O) will certainly cut down on your work during debugging. Type in your data or create your data file so that each data line has the following information: two integers: the beach number and the number of samples; followed by (on the same line): a series of numbers which are the number of organisms per 100 ml of water for each sample. These will eventually go into variables b_num, num_samples, and repeatedly (as you did, above) into num_orgs_per_100. eg for beach number 17 with 3 samples: 17 3 2500 3450 1825 You will eventually write a while loop in your coding (not yet) which will scanf values into b_num and num_samples. The while loop will be written to stop when the beach number is -17. The data could look like this:

```
18    3    2500   3450    1825
14    2    4800   6000
-17   0
```

Note that beach # -17 does not exist (hence the 0 samples). That line is there to signal the end of the file. We will get rid of it later.

g. Write a while loop which uses scanf to read in two values into b_num and num_samples. You will initially be using a negative value for the beach number (eg -17) as a sentinel to stop the iteration. Thus I, II, III, IV, and feof from the lecture notes are not applicable, at least at this point in the lab. Do not run this loop yet as it has no way of dealing with the trailing number of organisms in each data line. If you want to be sure what you have written works (not a bad idea to save LOTS of headaches later on), put a single printf in the while loop to output b_num and try it with a limited data set that does not have the trailing values: 18 3 14 2 -17 2

h. Inside a while loop place your "for loop" to read in the trailing numbers in each line (num_samples of them). For now these will be placed (repeatedly) into the single variable num_orgs_per_100. As before, after each value (num_orgs_per_100) is read in, print it out so you will be using the same for loop code as above. Here is how to write the "double looping". Note that the for loop control variable num_samples is read in prior to the for loop being executed. You will have to write a scanf to read the b_num and num_samples BEFORE the while loop; the while loop condition tests whether b_num is -17 (the "sentinel" value) or not; if b_num is not -17 the while loop body executes. In the while loop body, the for loop reads in and outputs each num_orgs_per_100 value. A copy of the same scanf as you wrote to precede the loop now reads a new b_num and num_samples from the NEXT data line. (Note that we have TWO statements we want to execute inside the while loop - the for and the scanf ... so they must be in a single compound statement which will form the while loop body.) After the scanf has input the first two values on the next line, the while loop test is performed again ... Note a three things: a) The first scanf statement (before the while loop) is executed only once. It initializes b_num for the while loop test, and it also gets num_samples which will be used to control the for loop doing the input for the remainder of that first data line. b) The while loop contains only a single statement - the compound statement { } containing the for loop and the second scanf statement. c) After the first iteration, the while loop condition is always testing the b_num value read in by the second scanf statement. The for loop (inside the while loop) will iterate (loop) num_samples times, each time reading in one value into num_orgs_per_100, and then in the second line inside the for loop printing it out. The while loop will thus read 18 and 3 into b_num and num_samples, respectively. The first iteration of the for loop will read and print out 2500; the second time through the for loop it will read and print out 3450; and the final ("num_samples") time through the for loop it will read and print out 1825. The while loop will next read 14 and 2 into b_num and num_samples, respectively. It will then input and print out the two bacteria count values 4800 and 6000, as you just went through immediately above. The while loop will continue on to read the next line of data and in this case

the b_num of -17 will cause the while loop to be finished. You now have a for loop running successfully inside a while loop.

i. Now, instead of printing out the num_orgs_per_100, they will be summed. This is done because we want the average of those values for each line. Remember that the sum variable (pick a good name like "total" for the summing variable) must be zeroed before each time you start to add up all the num_orgs_per_100. Just inside the while loop, zero your total variable right before entering the summing loop (your for loop). Change the printf statement in the for loop to a sum statement: total = total + num_orgs_per_100; After the for loop (still in the compound statement which is the body of the while loop, but BEFORE you read in num_samples for the next line - that MUST be the final statement in the while loop body) write the code to produce the average number of organisms per 100 ml for the line being processed. (You divide the sum for that data line by the number of samples in that line.) Put in a printf in the next line (before the second scanf) to output that average. Run and test that your code is producing the correct averages and stopping on the sentinel value.

j. There is still one job to do with the data for that beach. It must be determined whether the beach is to be closed or not. Right after the average has been output, you will need an if statement to decide whether the beach is to be closed or not. If the average is below 3500 print out the beach number and a short message that the beach is safe; if greater than or equal to 3500 print out the beach number and a short message that the beach is closed.

k. At present, you are using the sentinel value of -17 to stop the while loop. This is to be changed so that the while loop will stop, instead, when there are no more data lines in the file (or to be typed in if you are doing the tedious work of typing in the data again and again). Use one of the forms I, II, III, IV, or feof from the notes in Ch 5 to have the while loop stop when the end of your data file is reached. You will only have to change the scanf lines and the condition. (Forms II and IV may be the easiest: remove both your existing scanf's which read the b_num and num_samples, and they are replaced by a single scanf in the while condition which reads in b_num and num_samples.) You can also add several more lines to your data file at this point if you wish.

Test your program with the following dat files: data5.txt, erie.txt, huron.txt.

Exercise #2:

a. If your program is fully functional and you still have time to spare, add a report to count the total number of beaches in the file and how many are open, and how many are closed.

PART II: DISCOVERY ACTIVITIES

i. Using your text book, or an Internet search (do not ask friends or TA) or simply by experimenting with Quincy, explain in a few sentences the difference between these two statements: *if (x > 0)* and *while (x > 0)*.

PART III: LAB REPORT SUBMISSION

1. Submit the .c file for programming exercise 1.
2. Submit the text file containing the answer to the discovery question.
3. Submit on D2L/Brightspace under Lab #5. Submissions are due at the end of the lab session. You must submit your work before leaving the lab.

After the lab:

1. Review the steps you took to perform the various operations in the lab.
2. Try to use a do...while loop if you have not already in one of your programs.

Homework:

- On paper (no computer needed), do the following programming project (write the code by hand as you would on a test or an exam).

5. Let n be a positive integer consisting of up to 8 digits, d_8 , d_7, . . . , d_1. Write a program to list in one column each of the digits in the number n. The rightmost digit, d_1, should be listed at the top of the column. *Hint:* If n is 3,704, what is the value of the digit when computed by using

```
digit = n % 10;
```

Test your program for values of n equal to 6; 3,704; and 170,498.

- Show your homework to your lab assistant at the **beginning of next week's lab**.
- If you wish, you may try your solution with the computer to see if you got the correct solution (no need show the computer version).

Last modified: January 30 2020 14:52:11.