Lab Task 1.

Program Objective: Using sentinel-controlled loop

Program name. Updated multiplication table program

Problem Statement.

We have discussed in class, that a sentinel-controlled loop depends on user input and continues until the user input a particular (sentinel) value. We have also discussed that a do-while loop can be used to validate user input.

In this lab task, you have to modify the multiplication table program in Online Lab Task 1 (Warm-up Task 2), so that it first validates the input number, for which multiplication table is printed. The valid values are in the range 2-20 only. The program then prints multiplication table upto 5 terms, but does not exit. Rather, it asks the user if he wants to continue the table upto 5 more terms. If the user responds with 'y', then the next 5 terms are displayed. This process continues until the user inputs some other value than 'y'.

(**Hint:** you will have to nest the counter-controlled loop for printing the multiplication table inside the sentinel-controlled loop, which inputs user choice.)

Lab Task 2.

Program Objective: Using endfile-controlled loop

Program name. Average student score (batch mode)

Problem Statement.

Write a looping code segment that takes as input up to 20 integer scores from file **indata**, and outputs their average. If the file contains fewer than 20 scores, the segment should still output the correct average. If the file contains more than 20 scores, the additional numbers should be ignored. Be sure to consider what happens if the file is empty.

Lab Task 3.

Program Objective: Using general-conditional loop

Program name. Pair of divisible integers

Problem Statement.

Design an interactive input loop that scans pairs of integers until it reaches a pair in which the first integer evenly divides the second.
(**Hint**: This is an example of a general-conditional loop, so identify the loop termination condition first.)

Home Task.

Program Objective: Using endfile-conditional loop

Program name. Named students score percentage (batch mode)

Problem Statement.

Write a C program that computes student grades for an assignment as a percentage given each student's score and the total points. The final score should be rounded up to the nearest whole value using the **ceil** function in the **<cmath>** header file. You should also display the floating-point result up to 5 decimal places. The input to the program must come from a file containing multiple lines with the student's last name, score, and total separated by a space. In addition, you should print to the console "Excellent" if the grade is
greater than 90, "Well Done" if the grade is greater than 80, "Good" if the grade is greater than 70, "Need Improvement" if the grade is greater than or equal to 60, and "Fail" if the grade is less than 50. Here is an example of what the input file might look like:

Weems 50 60
Dale 51 60
Richards 57 60
Smith 36 60
Tomlin 44 60
Bird 45 60

Hint: Use endfile-controlled loop to read each line of data from file. In each line, the 2 numbers are read as integer variables, while, the name is read character-by-character using sentinel-controlled loop, where the sentinel-value is the space character ' '.

The output of your program should look like this:

Weems 83% .83333 Well Done
Dale 85% .85000 Well Done
Richards 95% .95000 Excellent
Smith 60% .60000 Need Improvement
Tomlin 73% .73333 Good
Bird 75% .75000 Good

Lab Assignment.

Program Objective:

- Develop input validation loop
- Develop sentinel-controlled loop
- Program separate functions for different sub-problems

Program name. Named students score percentage (batch mode)

Problem Statement. Write a program that inputs 2 numbers (n and r). Both inputs should be validated using a function int validateInput();, where valid values are positive integers. The function keeps on scanning a single integer value until a valid value is available. The function should then return this valid value. (**Hint**: use input-validation do-while loop format for this and develop the input validation function as done in online lab 2.)

After validating the inputs, the program shows the following input menu to the user. (**Hint:** you may use a separate void function for displaying these instructions only.)

Press 1 for multiplying numbers.
Press 2 for finding nCr.
Press 3 for finding nPr.

Press -1 for Exit.

The program should then perform the appropriate task according to the *choice* input by the user. Ignore validation of the *choice* input. (**Hint**: Use a sentinel-controlled loop format for this, where sentinel value is -1.)

(**Hint**: Formulae for computing combination and permutation:

- Comb = n!/(r!*(n-r)!)
- Perm = n!/((n-r)!)

Use a counter-controlled loop for computing factorial of n (n!). This is a running product pattern very similar to running sum, as discussed in lecture 1 on while loop.)