CS 1136 Laboratory Lesson #9b

Loops and Files

Important Information

This is your 9th lab lesson for the semester and will use the following guidelines.

Late submissions will get 0 points. If you save it as a draft and do not actually click on submit you will get 0 points. If you submit an empty file you will get 0 points. If you submit something other than the source files (for either the text files required or for the C++ source) you will get 0 points. Any other condition that results in incorrect or incomplete submissions will be graded as is and no changes after the due date will be allowed.

Anything you submit after the due date will be ignored.

Submissions must be made via eLearning. You can make multiple submissions and the final version you submit – before the deadline – will be used.

Note that there is an optional exercise. You can get up to 20 extra points for the optional exercise.

Part 1: Fun with output and loops

There are two files you can review. One covers nested loops – file **CS 1136 Laboratory Lesson 9b nested loop review.pdf**, and the other covers file input and output – file **CS 1136 Laboratory Lesson 9b file review.pdf**.

Problem description

Write a program called **Lab9b_Exercise1.cpp.** The main function should consist of a processing loop that will call an input function that reads in an integer value between 0 and 10. If the function returns back a value between 1 and 10 the main function should call a display function that will output that number of $ sign characters. If the input function returns back a 0 the loop should stop processing, otherwise the program should continue the loop and read in the next input value.

The input function reads in the input value and verifies that it is between 0 and 10. If it is not in this range you need to display an error message and prompt again for the input value. This must be done in a loop. You won't exit the input loop until the application n user enters in a number between 0 and 10 inclusive.

The display $ function will be called from main if the input value was between 1 and 10 inclusive. The function should then output that may $ characters on one line. You must use a loop to do this. You cannot just use 10 different cout statements. See the sample output below for examples.

Name the file Lab9b_Exercise1.cpp.

Here is an example input / output:

```
Enter the next number (1-10):
7 [Enter]
$$$$$$$
Enter the next number (1-10):
11 [Enter]
That is an invalid number.
Enter the next number (1-10):
-1 [Enter]
That is an invalid number.
Enter the next number (1-10):
10 [Enter]
$$$$$$$$$$
Enter the next number (1-10):
5 [Enter]
$$$$$
Enter the next number (1-10):
3 [Enter]
$$$
Enter the next number (1-10):
 2 [Enter]
$$
Enter the next number (1-10):
0 [Enter]
```

The file **CS 1136 Laboratory Lesson 9b nested loop review.pdf** contains an example of a program that outputs a rectangle made up of @ characters. You can use this to help you display a varying number of $ characters in your program.

Be sure to run the following tests:

Description	**Results / Actions**
Try a number less than 0	The program should display an error and ask the user to reenter the value. This should continue until a valid value is entered.
Try a number greater than 10	The program should display an error and ask the user to reenter the value. This should continue until a valid value is entered.
Try 1	The program should display one $
Try 10	The program should display: $$$$$$$$$$
Try other numbers between 1 and 10 (inclusive)	Your program should display the appropriate number of $'s

Description	Results / Actions
Various invalid and valid numbers	You program needs to continue to run until a 0 is entered.

1. Create an empty project with the name **Lab 9b Exercise 1 Project**.

2. Create your program with the name **Lab9b_Exercise1.cpp**

3. Make sure the following comments are at the beginning of your program. Replace **Place your name here** with your actual name.

```
// Lab 9b Exercise 1
// Display a row of $ characters based on user input
//
// Program by:     Place your name here
```

4. **Make sure you use meaningful names in your program. Also you should include comments to describe what your program is doing. You can use the problem description as a starting point for this. Put comments throughout your code.**

5. See the problem description above and the sample output for guidance on what is needed for the output of your application. The problem description also suggests a number of sample input values you should use for your application testing. You will need multiple test runs of your application to make sure your program is running properly.

6. If you are not sure how to output a varying number of $ characters you can look at the example program in the **CS 1136 Laboratory Lesson 9b nested loop review.pdf.** The sample code outputs a rectangle made up of @ characters. You can use this to help you display a varying number of $ characters in your program. The program outputs a varying number of rows and columns of characters, you are only looking at a varying number of (columns of) $ characters.

7. Test and debug your program. Try it with different, valid, values for the tests. Some sample run values are shown in the description above. Also run the application with invalid input values to check your error detection code and verify that your error messages are being displayed.

8. See the grading guidelines in **Part 3**. You will submit the C++ source file (.cpp file) to eLearning. You will submit the **Lab9b_Exercise1.cpp** file in **Part 5.**

Part 2: Reading from a file

Problem description

Write a program that reads a file containing an unknown number of integers and displays the percentage of the integers in the file that are zero, the percentage that are negative and the percentage that are greater than zero.

Name your program Lab9b_Exercise2.cpp. The user is not supposed to enter any data when the program is running.

(Use a simple text editor like NotePad to create an input file to test your program.) Unless you want to include the full path in your string literal, be sure to place the file in the same directory as your program.

Here is a sample output from a program that meets the requirements. This makes use of the file numbers.txt. This is included as part of your assignment on eLearning. You can use this or a file of your choice.

The processing in this program must be performed in functions. The main should only drive the work by calling the functions that actually do the work. You need to have the main function and, at least, 2 additional functions. Depending on how you write your functions you many need to pass parameters by reference.

Example:

```
There were 26.7% negative numbers.
There were 20.0% numbers equal to zero.
There were 53.3% numbers greater than zero.
```

1. You should create your Empty Project with the name **Lab 9b Exercise 2 Project.**

2. Create your source file and give it a file name of **Lab9b_Exercise2.cpp**.

3. Make sure the following comments are at the beginning of your program. Replace **Place your name here** with your actual name.

   ```
   // Lab 9b Exercise 2
   // Percentages of numbers input from a file.
   //
   // Program by:     Place your name here
   ```

4. **Make sure you use meaningful names in your program. Also you should include comments to describe what your program is doing. You can use the problem description as a starting point for this. Put comments throughout your code.**

5. See the problem description above and the sample output for guidance on what is needed for the output of your application. The problem description also suggests a number of sample input values you should use for your application testing. You may need multiple test runs of your

application to make sure your program is running properly.

6. Your application output does not have to match the output shown in the problem description, but all of the information shown above must be displayed in some way. Make sure you output the percentages and that they are between 0% and 100%.

7. Test and debug your program. Try it with different, valid, values for the input file. Some sample run values are shown in the description above.

8. You will submit the C++ source file (.cpp file) to eLearning. See the grading guidelines in **Part 3**. You will submit the **Lab9b_Exercise2.cpp** file in **Part 5**.

Part 3: Here is the general grading we will be using for Lesson 9b.

Note that you MUST submit working code with correct logic or points will be deducted. If you are having problems see your lab assistants, the tutors in the open lab and your instructor for CS 1136.

Lab 9b Exercise 1 is worth 35 points.

To get any credit you must upload the Lab9b_Exercise1.cpp file (the actual C++ source file).

Any syntax errors in your program will result in a minimum 5 point automatic reduction. You need to make sure you submit working programs.

Significant logic errors will result in a minimum 10 point reduction. Minor logic errors will result in a minimum 5 point reduction.

Not using meaningful variable names will result in a 5 point reduction. Failure to include comments will result in a 5 point reduction. There must be more comments that the required ones at the top of the application.

Note that the lowest grade you can get for exercise 1 is 0.

Lab 9b Exercise 2 is worth 65 points.

To get any credit you must have the Lab9b_Exercise2.cpp file (the actual C++ source file).

Any syntax errors in your program will result in a minimum 25 point automatic reduction. You need to make sure you submit working programs.

Significant logic errors will result in a minimum 25 point reduction. Minor logic errors will result in a 5-15 point reduction. Minor typos will result in a 5-15 point reduction

Not using meaningful variable names will result in a 10 point reduction. Failure to include comments will result in a 10 point reduction. There must be more comments than the required ones at the top of the application.

Note that the lowest grade you can get for exercise 2 is 0.

Part 4: Optional Exercise – This is worth 20 bonus points.

Write the source code for a program called Lab9b_Optional.cpp. Your program will use nested loops to print the pattern below to the screen. Remember to use descriptive variable names. You program will get the width of the base of the triangle from the user. You should only need a few cout statements to actually build the triangle. Depending on how you do this you may four or five cout statements, but you must not just have **77** cout statements. You should be figuring out how do this the programmatically for any positive value.

You will need to have several functions in addition to your main function. The main will call the other functions. You will need to have a function to read in the width of the base of the triangle. This function must ensure that the value of the width is between 4 and 80. If the width is not, the user must be asked to enter in a new value. This will continue until the user enters a value between 4 and 80, inclusive.

You will need another function to actually build and display the triangle.

Example output number 1:

```
Enter max width: 3 [Enter]
Invalid value. Value must be between 4 and 80.
Enter max width: 0 [Enter]
Invalid value. Value must be between 4 and 80.
Enter max width: 4 [Enter]
*
**
* *
****
```

Example output number 2:

```
Enter max width: 81 [Enter]
Invalid value. Value must be between 4 and 80.
Enter max width: 34 [Enter]
*
**
* *
*  *
*   *
*    *
*     *
*      *
*       *
*        *
*         *
*          *
*           *
*            *
*             *
*              *
*               *
*                *
*                 *
*                  *
*                   *
*                    *
*                     *
*                      *
*                       *
*                        *
*                         *
**********************************
```

Before you start this exercise you need to create a new empty project, **Lab 9b Optional Project**, with source file **Lab9b_Optional.cpp**.

Make sure you try some invalid widths and the minimum and maximum valid widths. Note that the triangle has a solid border, but is not filled in with *'s (that *'s are only on the borders with spaces everywhere else).

The tricky part of this exercise is figuring out when to output an * and when to output a space. You need to ask yourself – how can I check to see if the character is a border character or not?

You may want to output a solid triangle first before you modify your program to just print the border. You will get partial credit if you stop with a solid triangle.

Now go to **Part 5** to submit your source files.

Part 5: Submit your files to eLearning

1. Go to elearning.utdallas.edu and submit your files for lab assignment 9b. The following files need to be uploaded. You need to upload these in one attempt. You can submit multiple attempts, but each attempt needs to contain all of the files.

   Lab9b_Exercise1.cpp
   Lab9b_Exercise2.cpp
   numbers.txt

   Where the numbers.txt file is whatever file you read into your program for Lab 9b exercise 2.

   For the optional exercise you need to upload:

   Lab9b_Optional.cpp

2. **Make sure you submit all of your source files (the two or three C++source files). You need to make sure you do this BEFORE the deadline. Late submissions will NOT be accepted.**

3. You can make multiple submissions. Make sure you verify that the source files have been correctly submitted. You can go out to eLearning and download the files and verify that they are what is required (the two C++ source files -one for exercises 1 and one for exercise 2 and the optional third C++ source file). If they are not correct you can submit the correct files. You need to do this verification and resubmission before the due date. Updates submitted after the due date will not be graded.