**Submission of Your Work**

You need to prepare and submit ONE SINGLE MS Word document to Canvas (in your lab section) as LastName\_FirstName\_Labxy.doc. It must contain:

* Your NAME is the only item on page 1
* For ***each*** question:
  + Specify the question number, if more than one question.
  + Source code. Copy/Paste your final source code. You must include standard “comment header” even if code is provided. *Do Not* paste a snippet of your source code, it must be copy/pasted.
  + Initial test plan. After reading the requirements, but **before** beginning any coding, create the test case table, below, completed through column Expected Output. Include in your report.
  + Final test plan. Write your program then complete the **test table** with actual output results and include in your report *AFTER* your source code.
  + Output results. Paste in a snippet of output showing results for **every listed test case in your final test plan**, labeled with test case #

Test Table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |

* Add / delete rows from Test Table as necessary
* Modify column widths as necessary
* Test both valid and invalid input
* Test for every output expected
* If failure is an expected output and it happens then that test Passes
* Any test that fails means the program must be fixed so that it passes the test
  + Failing tests need a new test row, ie 1a, 1b, etc, showing corrections from original
  + Unless you don’t have time to finish your testing, there should never be a Fail in final test plan

Program:

Read in a file of supposedly positive integers (>0) from “integer.dat” (perform file validation – e.g. file exists, empty file, etc.) and create a doubly linked list of integers, adding values in descending sort order – print out any error messages to the screen and to a file “errorlog.txt” if the integer is <1 but continue processing the file through the end.

In addition, include as part of the list a doubly linked “sub-list” of only even integer numbers and only odd integer numbers. This means that there is a minimum of FOUR pointers for each node in the list – forward/backward full list, forward/backward even or odd list.

Write one function to print the entire list, even number list, and odd number list in either ascending or descending order. Use input parameter with value ‘A’ (ascending) or ‘D’ (descending) to tell which direction to print.

The main program, after reading in from the file, will also allow the user to add (remember must be >0) or delete integers from the list.

In all cases (both from file and from user), notify the user with an error message when:  
 number to add is already in the list  
 number to delete is not in the list

Have appropriate welcome and ending messages print to the screen, but NOT to the log file.

Sample node has data, prev, next, prevEvenOdd, nextEvenOdd

8

0

5

2

Head

0

1

0

HeadEven

0

0

0

HeadOdd