CS-200 Fall 2015

LAB 15

MAXIMUM TIME ALLOWED 3: Hours

TAs has been asked to deduct marks if you ask very basic questions. Also if you ask TAs for solving basic syntax errors, you marks will be deducted. It is time that you learn to code independently and learn to understand the errors on your own.

- Always keep a notebook and a pen with for any programming assignment
- First understand the problem, plan how you will tackle it on the notebook
- Write a rough pseudo code for your solution to the problem and then start to code.

It will be a lot easier and time saving; otherwise you will hit a wall right in the middle of writing code because you hadn't thought about the problem fully.

You can take help from lecture slides uploaded on LMS.

Also you will be marked for

- proper variable naming (10 mark deduction if this is not done)
- proper indentation (10 mark deduction if this is not done)

Indentation helps make the code readable and easy to understand. A code which is not indented is just like a text with no full stops, commas, paragraphs or general punctuation whatsoever, it is almost impossible to read.

Code has to be indented after every curly bracket '**{**'. And whenever those brackets are closed '**}**', the remaining code is not indented. Look at the following.

```
int main(){
        // I will write my code here because of a curly bracket
        // ok let me open another curly bracket
        {
                // now I am indenting more than before
        }
        // back again
}
```

Use tabs for indenting your code so that it is readable. Open a cpp file by writing -

nano –i task1.cpp

This allows Nano to do some indenting on its own. But you have to indent each first line after a curly bracket by yourself; it then remembers the level of indentation for the next lines. Perhaps this may be more useful when you write code with many levels of indentation!

All grading will be done in lab so finish your lab work in time! No grading after the lab is over and upload your work.

You can take help from slides and the book

Task1 [50 points]:

Convert functions in "***merge.h***" to template based functions. Currently all three of them are defined for ***int*** only. It should every data type with an operated< defined for it.

void merge(int arr[], int l, int m, int r)

void mergeSort(int arr[], int l, int r)

void printArray(int A[], int size)

Use your modified ***"merge.h"*** to sort double, char, and string arrays in "***main.cpp".*** You have to change a little to determine correct size of the array. Currently it is only calculating size of an **integer array.**

TASK-2 (50 points)

Write a function **"2StackSort".** This function will take an unordered stack and return an ordered stack. But I can only use one more stack and one temporary variable and some counter variables.

std::stack<T> 2StackSort (std::stack<T> st)

Read "Cpp_STL_ReferenceManual.pdf" there is a STL class Stack in C++. Use it to create Stacks for your function. This student objects for each line in the data file and insert it into a vector.

Format of name in Lab submissions

<Student-ID>-Lab<#>-<TA-ID>.zip

For example 18100075-Lab4-Huzaifa.zip means this is the lab submission of 18100075 for Lab4 checked by Huzaifa. This way it will be easier for students and TAs to figure out the TA in case of any problems.