The due date for this assignment isThursday, 10/6th, at 11.59pm*.* This assignment is worth 12.5% of your final grade.

Any sign of collaboration will result in a 0 and being reported to the Graduate Academic Integrity Board. The programming assignment will be done individually. No collaboration is allowed between students. No code from online resources is allowed to be used besides the code that I will share with you. Late submission policy described in the syllabus will be applied.

You are given an integer vector which is represented by *int\** an array of integers and its dimension *n* as a separate parameter. We are interested in sorting arrays of integer vectors according to a pre-defined notion of vector length. You therefore are given the function *ivector\_length(v, n)* that computes and returns the length of vector *v* with dimension *n* as 

You are given a naive (and very inefficient) implementation of insertion sort for arrays of integer vectors.

**Questions (100 points)**

1. Develop an improved implementation of insertion sort for integer vector (*insertion\_sort\_im*) that precomputes the length of each vector before the sorting. Keep in mind that the vectors are sorted according to their length (see *ivector\_length* function). You can test the correctness of your sorting algorithm using the provided *check\_sorted* function.

2. Implement a merge sort for an array of integer vectors. For this implementation of the merge sort, as is the case for the improved insertion sort algorithm, you should precompute the length of the vectors before the sorting, and the sorting is done according to the vector lengths. Test the correctness of your merge sort implementation using the provided *check\_sorted* function.

3. Measure the runtime performance of insertion sort (naive and improved) and merge sort for random, sorted, and inverse sorted inputs of size *m* = 10000; 25000; 50000; 100000; 250000; 500000; 1000000; 2500000 and vector dimension *n* = 10; 25; 50. You can use the provided functions *create\_random\_ivector*, *create\_sorted\_ivector, create reverse\_sorted\_ivector.*

Repeat each test a number of times (usually at least 10 times) and compute the average running time for each combination of algorithm, input, size *m*, and vector dimension *n*. Report and comment on your results.

Sample test case:

Consider input [ [4,2,1], [9,2,1], [1,2,1] ],  
Output should be [[1,2,1], [4,2,1], [9,2,1]]

Explanation: 1+2+1 < 4+2+1 < 9+2+1

Sorting occurs according to sum of numbers at that index as shown above.

Vector just means an array. Eg. [4,2,1] is a vector

Length means sum of numbers in that vector. Eg. Length of vector [4,2,1] is 7

Remarks:

* You are not allowed to use code from online resources. Your submission will be tested against that, and will receive a 0, and a report to the Graduate Academic Integrity Board if it is detected.
* Please submit one single "Lastname\_firtsname.zip" file which will contain a pdf file which will be the report and a folder that contains your code. All the readings and graphs should be inside the word/pdf file. No extra excel file should be submitted.
* Code folder should have same number of code files with same name as the original code folder provided. Don't submit only "sort.cpp". If you are provided with 7 files, 7 files should be submitted with same name. ".o" files and "./hw1" binary files are not supposed to be submitted. Extra helper code files and shell scripts used should be excluded as well.
* Format for report has been provided in L03 slides at slide no. 51. Please make your reports according to that. You do not need to write or explain your code lines.
* Try to create graphs and plots similar to the ones provided in L03 slides at slide no.52. Don't ignore the lines "C1nlogn" and "C2nlogn"(for merge sort) as they are required in the graphs for that particular algorithm. They are to be included in upcoming assignments as well to see how similar your readings are according to the function graph. You need to randomly select values of C1 & C2 by which your C1nlogn(for merge sort) value will be close to your readings and on the same scale. Example: if your runtime for a particular input is 100, select value of C1 and C2 such that C1nlog and C2nlog are around 80-120.
* Note - In these equations "n" will be actually "m" since m is the number of vectors, so it will be "C1mlogm".
* Your report has to be typed, and submitted in a pdf file. In the report, you should include
  + You must present the running time result table for each *m, n, direction,* and sort.
  + You must provide plots showing the running times in relation to the input size parameters. Use a scatter/line plot (no bar plots) for the better presentation(similar to sample graph provided in lecture 3 slides).
  + You must discuss the theoretical running time and the behavior of the tested running time. You also must fit the result running time to the theoretical model.
  + You must discuss and evaluate your work. Report and comment means that you have to analyze and interpret your findings properly. What do the experiments tell you?
* You might have to adjust the value for *n* and *m* depending on your computers speed, but allow each test to take up to **a couple of minutes**.
  + In this case, you must provide the threshold of *m* and *n.*
* Start with smaller values of *n* and *m* and stop if one instance of the algorithm takes more than 5 min to complete (the insertion sort implementations will hit that limit early on).
* The programming, testing and the experimentation will take some time. Start early.
* Feel free to use the provided source code for your implementation. You have to document your code.
* A Makefile is provided to build the code in the Virtual Box provided.
* Your code has to compile, and run at the Virtual box shared with you in order to be graded.
* Your methods should be added in the sort.h/sort.cpp files. You must keep the same file structure, makefile that is provided in the skeleton code.
* Your code MUST compile and run on our machines. If not, minimal marks will be given. To ensure this, before submission please run your code on “https://www.onlinegdb.com/” as the TA’s will be running your code on this website with “C++” selected. If your code is to be run on “C++14” or “C++17” then you need to mention it in submission comments while submitting.
* You do not need to change anything in "random\_generator" and "timer" files, please refrain from changing them.