HOMEWORK 1

100 points
DUE DATE: September 15th 11:59pm.

This homework contains 4 problems.

Warning: For any homework assignment that contains a programming segment please read the following very carefully.

Your code must compile and run on Black server.If your code does not work on Black server as submitted the grade for that problem is 0 regardless of its quick fix(es).Always test your code on Black, even if it is incomplete, make sure to get your code to compile and run on our Servers.

General Definition of Problem 1

Suppose you have a group of N numbers and would like to determine the k^{th} largest. This problem is a very common programming problem. Since you have a background in programming, this should not be too difficult to solve. Write a program to solve this problem.
There are several ways to approach to solve this problem. In this homework You will apply 3 different ways to solve this problem

1. Approach 1: One way to solve it to read all N numbers into an array (or a vector), sort the array in decreasing order by some simple sorting algorithm (bubble sort), and return the element in position k. We will call this method: **select1**
2. Approach 2: Since we are solving a problem to find the k^{th} largest element from an array[0…n-1] (or a vector v). Better approach might be to use of a temporary array (or a vector) and store the first k elements into a temp array (or a vector) and find its min element(you can use std::min). Remaining element from the array[k] to array[n-1] will be checked against this min. When the program finds that array[i]th element is greater than the current min, then it must override this value in that **temparray** where the min value was found. A new minimum value in the temp array must be calculated each time the minimum value is overridden. Once every element has been checked, you should have the k largest elements in the temporary array, but they will not be sorted. Sort the temporary array in decreasing order and the k^{th} largest element will be at index k-1. This method is better when n is large and k is small, because it only involves sorting k elements instead of all n elements.

See explanation below to understand this selection problem. This example contains a very small data file(unsorted) to demonstrate the point. The program chooses k^{th} largest val(using Approach1 or Approach2). Value for K will be generated randomly

```
Small Input file
19459
14309
20013
15680
23348
31998
11548
772
6771
25819
---------------
K th val is :  4
Print the kth largest element
4 th largest value in array -->  20013
```

```
Small Input file
19459
14309
20013
15680
23348
31998
11548
772
6771
25819
---------------
K th val is :  8
Print the kth largest element
8 th largest value in array -->  11548
```

With your requirements document you are given:

1. An incomplete KthLargest.h
2. main.cpp
3. makefile And
4. inputfile

Please do not modify: main.cpp, make and inputfile for Problem1.

Problem 1

Complete KthLargest.h. Save your work.
You will be completing the following methods:

```
void sort(vector<Comparable> &vec)
void sortDec(vector<Comparable> &vec)

void  select1(vector<Comparable>  nums, int kthval)

void  select2(const vector<Comparable> &nums, int kthvalue)
```

sort method: uses a simple sort such as bubble. It sorts data in increasing order.
sortDec: uses a simple sort such as bubble. It sorts data in **decreasing** order.
select1: Uses the sort method for finding the k^{th} largest value in a given array(or vector)
select2: Uses a temp array for finding the k^{th} largest value in a given array.
Feel free to write your own print method to display your result. You can call your print method from select1 or select2. You can use the size function if you are using a vector.

For the documentation:
Please place a comment box at the top of KthLargest.h stating your name, hw 1, and its due date. See image below for guidance:

```
/*
 * KthLargest.h
 *
 *      Created on: September 6th 2016
 *      Author: Sebnem Onsay
 *      Homework 1
 *      Due: September 15th
 */

#ifndef KTHLARGEST_H_
#define KTHLARGEST_H_


#include <iostream>
#include <ctime>
#include <cmath>
#include <vector>
#include <algorithm>
using namespace std;

template <typename Comparable>
void  simpleprint(const vector<Comparable> &nums)
```

Compiling using make file and testing work on Black Server

Test your code using main.cpp. Please do not modify the main file when you are testing for Problem1.
Make sure that your code compiles and run on Black Server (C++ 11).
For testing your code you will need to use the make file. This file simply compiles your code. Here are the steps on how to use the make file:

```
[onsayse:~/CSE331_F16/HOMEWORK/HW1]$ make
g++ -std=c++11 -g -O3 -c main.cpp
g++ main.o -o kthsearch
[onsayse:~/CSE331_F16/HOMEWORK/HW1]$
```

kthsearch executable is created after make command

```
[[onsayse:~/CSE331_F16/HOMEWORK/HW1]$ ls -l
total 1904
-rw-------@ 1 onsayse  staff  411239 Jul  7 14:50 HW1_Collection.pptx
-rw-------@ 1 onsayse  staff  119397 Jul 18 18:41 HW1_Requirements.docx
-rw-r--r--  1 onsayse  staff    3779 Jul 18 18:44 KthLargest.h
-rw-r--r--  1 onsayse  staff     142 Jul 18 18:59 Makefile
-rwxr-xr-x@ 1 onsayse  staff  133142 Sep  8  2015 input.in
-rwxr-xr-x  1 onsayse  staff   37108 Jul 18 19:00 kthsearch
-rw-r--r--  1 onsayse  staff    2466 Jul 18 18:55 main.cpp
-rw-r--r--  1 onsayse  staff  243716 Jul 18 19:00 main.o
drwxr-xr-x  3 onsayse  staff     102 Jul 18 18:57 sol
-rw-------@ 1 onsayse  staff     162 Jul 18 11:53 ~$1_Requirements.docx
```

now run your program

```
[[onsayse:~/CSE331_F16/HOMEWORK/HW1]$ ./kthsearch
Approach 1: Using Bubble Sort
Print the kth largest element
8 th largest value in array -->  32756
Size of the vector : 20000
Bubble Search method  took 667 milliseconds

Approach 2: Using Temp Array
Print the kth largest element
8 th largest value in array -->  32756
Size of the vector : 20000
26516 nanoseconds
[onsayse:~/CSE331_F16/HOMEWORK/HW1]$
```

Problem 2

In this segment you are allowed to modify the **main.cpp** to test your methods for different input sizes since you will be running it for different data sizes to collect data.
Draw a table showing the running time of your program for select1 and select2 methods for the following N values 2000,4000,6000,8000,10000,12000,14000. Use any program to be able chart your data. Copy your chart and paste it on a text document and convert this document to PDF for submission. We will only accept PDF format. Make sure to convert your document to PDF prior to submission.

Problem 3

Using Proof by Induction: Prove that the derivative of x^n w.r.t. x is nx^{n-1} for $n \geq 1$ using

- the chain rule, and
- $(x)' = 1$

You can complete this problem on paper or using a text editor with equation editor. However, if you are using a paper, make sure to place your name on the top of the paper, place the Problem number and scan your paper to turn it into PDF for submission. If you are using a text editor, make sure to convert your file into PDF prior to submission.
Use Lecture 0 Math review slides to help you solve this problem.

Problem 4

Using Proof by Induction: Prove that

$$\ln(n!) \leq n\ln(n)$$

for integer values of $n \geq 1$

- When $n = 1$:

$$\ln(1!) = \ln(1) = 0 = 1 \cdot \ln(1)$$

- Assume that:

$$\ln(n!) \leq n\ln(n)$$

You can complete this problem on paper or using a text editor with equation editor. However, if you are using a paper, make sure to place your name on the top of the paper, place the Problem number and scan your paper to turn it into PDF for submission. If you are using a text editor, make sure to convert your file into PDF prior to submission.
Use Lecture 0 Math review slides to help you solve this problem.

Homework 1 Deliverables:

The following files must be submitted via Handin no later than September 15th 2016 by 11:59pm

1. **KthLargest**.h

2. **Problem 2: DataTable with a Graph in PDF format (Table.**PDF)
3. **Problem 3 and 4 in PDF format.**