HOMEWORK 2

100 points

DUE DATE: September 22nd 11:59pm.

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.

This homework contains 4 problems.

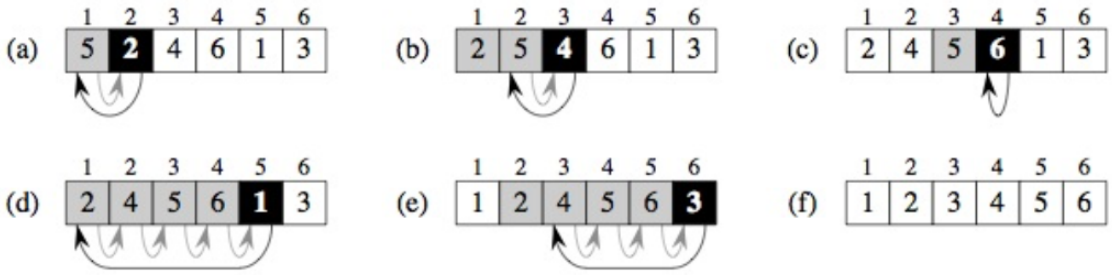
With your requirements document you are given:

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

Please do not modify: main.cpp,make and input file.

Problem 1:

Using the figure below as a model, demonstrate the operation of INSERTION-SORT on the following array A= {31,41,59,26,41,58}. (Use the pseudocode provided in class to demonstrate the algorithm)



You can complete this problem on paper or using a text editor with drawing tools. However, if you are using a paper, make sure to place your name on the top of the paper, place the title as Problem 1 draw your array and demonstrate your INSERTION-SORT. 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.

Problem 2:

In order to complete this problem, you are provided with a template named as **SearchSort.h**. Use this h file to complete this problem.

a) Convert the following pseudo code for INSERTION-SORT into a C ++ program.

```
INSERTION-SORT(A)
for j ← 2 to n
    do key ← A[j]
       ▷ Insert A[j] into the sorted sequence A[1 . . j − 1].
       i ← j − 1
       while i > 0 and A[i] > key
           do A[i + 1] ← A[i]
              i ← i − 1
       A[i + 1] ← key
```

b) Re-write the INSERTION-SORT in C++ to sort into decreasing order instead of increasing order.

```
void insertion_sort(vector<Comparable> &A);
void insert_sort_dec(vector<Comparable> &A);
```

Problem 3:

You have already completed a linear search during Hw 0, now you will analyze your algorithm. In short terms Linear search (using an array or a vector) is:

Input: A sequence of n numbers

$A = \left(a_1, a_2, \ldots, a_n \right)$ and a value v.

Output: An index i such that:

$v = A[i]$

or the special value *NIL if* v does not appear in A.

a) Write a pseudo code for your Linear Search (Using the same style shown in Problem 2 and also in class)

Name your array as A and you can use the length property of your array for example

h ← length[A]

Place a line number on each line of code for example

1. h←0
2. while h<= length[A] …
3. …..

b) Analyze your algorithm. When analyzing your code place a cost and times column next to your pseudo code as demonstrated on examples in class. For example:

	Cost	Times
1. H ←0	c1	1
2. while h<= length[A] …	c2	….
3. …..		

Compute the T(N).

a. What is the best case? Describe with a simple sentence and explain in terms of Θ.
b. What is the worst case? Describe with a simple sentence and explain in terms of Θ.
c. What is the average-case? Describe with a simple sentence and explain in terms of Θ.

You can complete this problem on paper or using a text editor with drawing tools. However, if you are using a paper, make sure to place your name on the top of the paper, place the title as Problem 1 draw your array and demonstrate your INSERTION-SORT. 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.

Problem 4:

For each of the following four program fragments, give an analysis of the running time in terms of Big-O. Your answer will be a very short statement → T(n) = O(?) Your algorithm run time analysis should only contain the highest term when writing in terms of Big O.

For example:

T(N)=O(n^3+5n^2) would be inaccurate, it should be simplified (dropping the lowest terms) as:

T(N)=O(n^3)

(a)
```
Sum = 0;
for i = 1 to N do
  Sum = Sum +i;
```

(b) Sum = 0;

for i = 1 to N do

for j = 1 to N do

for k = 1 to N do

Sum = Sum + 1;

(c) Sum = 0;

for i=1 to N^2 do

for j = 1 to N do

Sum = Sum + 1;

You can complete this problem on paper or using a text editor with drawing tools. However, if you are using a paper, make sure to place your name on the top of the paper, place the title as Problem 1 draw your array and demonstrate your INSERTION-SORT. 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.

Homework 2 Deliverables:

Use the make file provided for you to test your program prior to submission. **Please note that in main.cpp, method calls were commented out to keep the compiler quite, make sure to uncomment them when you are ready to test your code.**

Reminder: Your code must compile on our Server.

```
[[onsayse:~/CSE331_F16/HOMEWORK/HW2/sol]$ ls -l
total 200
-rw-r--r--  1 onsayse  staff    144 Aug  3 22:53 Makefile
-rw-r--r--  1 onsayse  staff   1378 Aug  3 22:44 SearchSort.h
-rw-r--r--  1 onsayse  staff    134 Jul 21 17:03 input_real_small.in
-rw-r--r--@ 1 onsayse  staff  83742 Jul 19 12:53 linearsearchproblem.pdf
-rw-r--r--  1 onsayse  staff   3516 Aug  3 22:40 main.cpp
[[onsayse:~/CSE331_F16/HOMEWORK/HW2/sol]$ make
g++ -std=c++11 -g -O3 -c main.cpp
g++ main.o -o searchsort
[[onsayse:~/CSE331_F16/HOMEWORK/HW2/sol]$ ls -l
total 728
-rw-r--r--  1 onsayse  staff     144 Aug  3 22:53 Makefile
-rw-r--r--  1 onsayse  staff    1378 Aug  3 22:44 SearchSort.h
-rw-r--r--  1 onsayse  staff     134 Jul 21 17:03 input_real_small.in
-rw-r--r--@ 1 onsayse  staff   83742 Jul 19 12:53 linearsearchproblem.pdf
-rw-r--r--  1 onsayse  staff    3516 Aug  3 22:40 main.cpp
-rw-r--r--  1 onsayse  staff  230188 Aug  3 22:57 main.o
-rwxr-xr-x  1 onsayse  staff   36468 Aug _3 22:57 searchsort
```

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

1. **SearchSort**.h
2. **Problem1 and Problem 3 Problem 4 in PDF format**