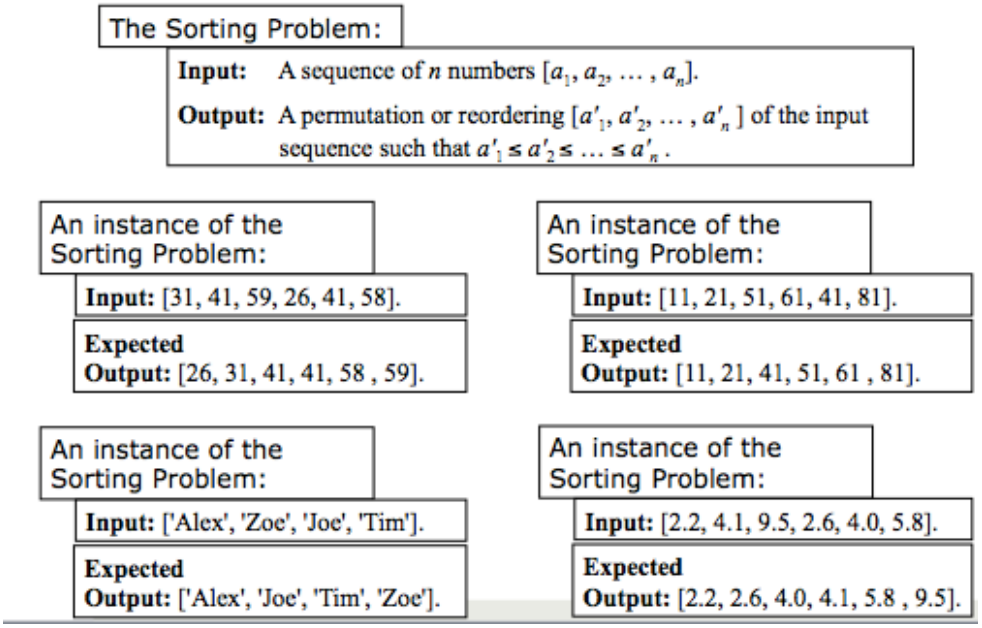
Sorting

In this assignment you will implement a first solution to the sorting problem.
The problem is formally specified in your textbook as follows:



The Sorting Problem:

Input: A sequence of n numbers $[a_1, a_2, \dots, a_n]$.

Output: A permutation or reordering $[a'_1, a'_2, \dots, a'_n]$ of the input sequence such that $a'_1 \leq a'_2 \leq \dots \leq a'_n$.

An instance of the Sorting Problem:

Input: [31, 41, 59, 26, 41, 58].

Expected Output: [26, 31, 41, 41, 58 , 59].

An instance of the Sorting Problem:

Input: [11, 21, 51, 61, 41, 81].

Expected Output: [11, 21, 41, 51, 61 , 81].

An instance of the Sorting Problem:

Input: ['Alex', 'Zoe', 'Joe', 'Tim'].

Expected Output: ['Alex', 'Joe', 'Tim', 'Zoe'].

An instance of the Sorting Problem:

Input: [2.2, 4.1, 9.5, 2.6, 4.0, 5.8].

Expected Output: [2.2, 2.6, 4.0, 4.1, 5.8 , 9.5].

Insertion Sort

Insertion Sort is an algorithm that solves the sorting problem. It works the way many people sort a hand of playing cards.

We start with an empty left hand and the cards face down on the table. We then remove one card at a time from the table and insert it into the correct position in the left hand.

To find the correct position for a card, we compare it with each of the cards already in the hand, from right to left.

Notice that, at all times, the cards held in the left hand are already sorted.

The Algorithm

Make sure you understand the following algorithm. It will be the basis for this assignment and the running time analysis that we will do later on.

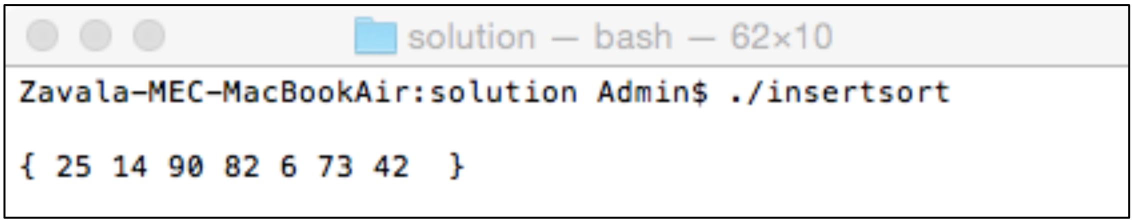
```
INSERTION-SORT(A)
1  for $j = 2$ to $A.length$
2      $key = A[j]$
3      // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.
4      $i = j - 1$
5      while $i > 0$ and $A[i] > key$
6          $A[i+1] = A[i]$
7          $i = i - 1$
8      $A[i+1] = key$
```

The Starter Code

Download the starter code for this project (*insertsort.cpp*) and take a look at the code. You should notice that the insertionSort function is empty. You will write the code for it later on. First, you will complete other tasks.

TASK 1. Write the code for the printArray function. The function takes an array and the size of the array as arguments. The function should print to screen the contents of the array it receives as argument. The contents of the array should be printed on a single line.

Run the program. You should see array A, which is created in the first line of the main function, displayed to screen:



```
Zavala-MEC-MacBookAir:solution Admin$ ./insertsort

{ 25 14 90 82 6 73 42  }
```

TASK 2. Write the code for the isSorted function. It should return *true* if the items in the array it receives as argument are sorted (in ascending order), and *false* otherwise.

Uncomment the first block of comments in the main function and run the program. The isSorted function should pass the test (you should not see any error message displayed).

TASK 3. Before you write the insertionSort function, take a look at the insert function and answer the following questions:

QUESTION 1 Which lines from the INSERTION-SORT algorithm is the function implementing?

QUESTION 2 A) How would you describe what the insert function does?
B) What assumption is made about the array that it takes as argument?

For task 3, you must make use of the insert function to sort array A manually. That is, you must make as many calls to the insert function as needed to have the array sorted. Do not use any loops.

Uncomment the second block of comments in the main function and run the program. Your program should pass the tests (you should not see any error message displayed).

TASK 4. Implement the insertionSort function according to the INSERTION-SORT algorithm. Make use of the insert function.

Uncomment the third block of comments in the main function and run the program. Your program should pass all tests (you should not see any error message displayed).

TASK 5. Create an array with 10 numbers of your choice in random order and call it *arrayLatName*, where you will substitute *LastName* for YOUR last name. For example, my array would be called arrayZavala. Then, call the insertionSort function to sort the items in your array. Next, use assert to test that the array is now sorted (using the isSorted function). Finally, you should print your sorted array (using the printArray function).