CS1010 Programming Methodology
Week 10: Searching and Sorting

> Being ignorant is not so much a shame, as being unwilling to learn.
> ~ *Benjamin Franklin*

To students:

Many programs for this discussion can be downloaded from cs1010 account. For example, to copy **Week10_Q1.c**, you can type:

```
cp  ~cs1010/discussion/Week10_Q1.c  .
```

I. Exploration (non-examinable)

The questions in this section are meant for your self-study. **They are not likely to be discussed in class**, as we expect you to explore such additional knowledge on your own.

1. **Timing your program.**

   Now that you have learned arrays, which allow you to hold a large amount of data, and algorithms with different running time complexities, you may be interested in timing certain parts of your program.

   The following program **Week10_Q1.c** illustrates how to time a *for* loop, using the **clock()** function in <time.h>. The value returned by **clock()** is measured in microseconds (10^{-6} second). Download **Week10_Q1.c** from cs1010 account and test it.

```
#include <stdio.h>
#include <time.h>

int main(void)
{
   int start, finish, i;

   start = clock();
   for (i=0; i<100000000; i++)
      ;
   finish = clock();

   printf("Difference = %.2f sec.\n",
          (finish - start)/1000000.0);
   return 0;
}
```

2. Choose the Selection Sort or Bubble Sort program and run it with arrays of different sizes, such as 1000, 2000, 4000, 8000, 16000. Verify that as the size of the array doubles, the time it takes to sort the array is roughly quadrupled, providing empirical evidence that the algorithm is quadratic in running time complexity.

II. Problem Solving with Sorting

3. We illustrated sorting algorithms using integer arrays in class. Determining whether one element, say a[i], is smaller than another, say a[j], is simply done by comparing a[i] with a[j] (e.g., `if (a[i] < a[j])`).

   What if the array elements are more complex (for example, each element is a string, or a structure comprising of more than one component – we will cover structures later), or the comparison criterion is more complex?

   Suppose you want to sort an integer array of 6 elements in increasing order of the first 3 digits of each element, how would you modify the selection sort algorithm in the given **Week10_Q3.c** program?

   A sample run is shown below:

```
Enter 6 values:
12345
9870
32
5555555
801784
729
After sort:
32 12345 5555555 729 801784 9870
```

Download skeleton **Week10_Q3.c** from cs1010 account

4. **Enhanced Bubble Sort**
   As mentioned in lecture, the Bubble Sort can be enhanced. If you detect no swap in a pass, it implies that the array is already sorted. Write an enhanced Bubble Sort program **Week10_Q4.c**, in which the sorting terminates after a pass with no swap.

   The running time of your enhanced Bubble Sort is sensitive of the initial order of the data in the array. When does the best case occur? When does the worst case occur?

   (There are other variants of Bubble Sort, such as Bidirectional Bubble Sort, also known as Cocktail Sort or Shaker Sort. Check out the Internet for details if interested.)

Download skeleton **Week10_Q4.c** from cs1010 account

5. **Insertion Sort**
   Insertion Sort is another basic exchange sort besides Selection Sort and Bubble Sort. Refer to the PowerPoint file in the IVLE workbin → "Discussion" for the Insertion Sort algorithm.

   Implement Insertion Sort on an integer array.

6. **Anagrams**
   An anagram is a rearrangement of all the original letters in a phrase, disregarding any non-letter characters. For example "A decimal point" = "I'm a dot in place". The website http://www.anagramsite.com/ contains a list of anagrams, some of which are rather interesting.

   There are a few approaches to check whether two phrases are anagrams of each other. Write an algorithm of the solution that uses sorting, and implement it as **Week10_Q6.c**. Test your program with the anagrams in the above website.

   You may assume that a phrase contains at most 60 characters.

   A sample run is shown below.

```
Enter 1st phrase: George Bush
Enter 2nd phrase: He bugs Gore
They are anagrams.
```

Download skeleton **Week10_Q6.c** from cs1010 account

7. [CS1010 AY2010/2011 Semester 1 Exam, Q7] Certificate of Entitlement

The Certificate of Entitlement (COE) is designed to limit the number of vehicles on the roads by requiring potential car owners to first obtain the right to buy a vehicle. Each month, the number of available COEs is made known and people who wish to buy a vehicle will submit their COE bids to a bidding system.

Suppose the number of available COEs for a particular month is N. At the end of the bidding cycle, we obtain the N^{th} highest bid as the COE Candidate Price. For example, if there are 4 COEs available for the month of March and the bids received are {1, 100, 50, 2, 8, 10000, 1000, 2, 1000, 10010}, then the COE Candidate Price is the 4th highest bid, which is $1000. This is also the COE Final Price.

In the event that the number of bids that are greater than or equal to COE Candidate Price is more than N, then the COE Final Price is set to the next highest bid that is greater than COE Candidate Price.

For example, if there are 4 COEs available for the month of April and the bids received are {1, 100, 2000, 2, 8, 10000, 1000, 2, 1000, 10010}, then the COE Candidate Price is $1000. However, there are 5 bids that are more than or equal to $1000. Hence, the COE Final Price for April is the next highest bid (you may assume that you can always find one) that is more than $1000, which is $2000.

Write a complete C program that takes as input the number of available COEs for the month and the list of bid amounts (terminated by a zero bid amount), and outputs the COE Final Price for that month. Bid amounts are integers. You may assume that there are no more than 5000 bids received per month.

Two sample runs are shown below.

```
Number of available COEs: 4
Enter bids: 1 100 50 2 8 10000 1000 2 1000 10010 0
COE Final Price this month is $1000
```

```
Number of available COEs: 4
Enter bids: 1 100 2000 2 8 10000 1000 2 1000 10010 0
COE Final Price this month is $2000
```