60-140 INTRODUCTION TO ALGORITHMS AND PROGRAMMING I

Dr. Ziad Kobti

Lab 8: Search and Sort

Objectives:

1. Learn how to code a linear search.
2. Learn how to code a binary search.
3. Learn how to do a bubble sort.

Instructions:

Recall in the previous lab you learned how to create a directory, change into it, and view its contents. (i.e. the UNIX commands you need to understand here are `mkdir` *`directory_name`*, `cd` *`directory_name`*, and `ls`)

In this lab you are asked to create one complete C program that includes several functions, but all inside the same source code file named **Lab8_q.c** in a directory called **lab8** which should be located inside the **cs140** directory you have already created in the first lab. The reason for this is to keep your files organized.

At the end of this lab class, you will have 1 source code file saved in a folder called lab8 under the cs140 folder. Recall that for the purpose of evaluating and grading your work by the GA's, you would need to create one script file which holds the source code, compilation result and the output of all programs. The process of creating such a file is described below:

```
>> script lab8.txt
>>  cat lab8_q.c     // will show and add the source code for the question to the script file
>>cc lab8_q.c   // will compile lab8_q.c
>>./a.out        // will run the program and appends its output to lab8.txt file
>>exit          //closes the script file. Your work is ready to be graded!
```

Submission:

Your lab is graded either at the end of your current lab class (or at the very beginning of your next regular lab class without penalty). Late labs without a valid excuse (eg. illness) receive 0. You are to present your code on the computer in the lab to your lab instructor to receive a grade as follows:

0= not satisfactory or no documentation; 1=incomplete; 2=complete and well documented

Lab question

In the previous lab you were asked to write a single C program that creates a one dimensional array of size MAX (to be declared as a declarative constant using directive `#define MAX 10`). Reuse the functions: InitArray, PrintArray, GetNumber, PopulateArray and SearchArray as much as possible here.
In this program you are asked to code a function called BinarySearch that will search the array for a given key, returning its index position if found, -1 otherwise. Note that in order to use BinarySearch your array needs to be sorted. Therefore, you will need to first write a function called BubbleSort to sort the array.
The structure for this C program is similar to this (modify as needed):

```
int main()
{
      //array A to be locally declared here

      int size = MAX;  // note: actual size used can be less than
                       // MAX memory allocation

      // populate the array with a number from the user between 0-10
      InitArray(A, size, GetNumber(0, 10));
      PrintArray(A, size); // to verify the output
      PopulateArray(A, size); // randomly generate values 0 to MAX/2
      PrintArray(A, size);
      BubbleSort(A, size, 1); // 1=ascending, 2=descending sort order
      PrintArray(A, size);
      BinarySearch(A, size, GetNumber(0, 10)); // search for a value
and display the results of your search. (put it in a printf)

      …

      Return 0;
}
```

List of function:

BubbleSort
This function takes as parameters the array, its size and an order flag. The flag can be 1=ascending, 2=descending order. The function sorts the array accordingly. Has no return.

BinarySearch
This function takes as parameters the array, its size, and a key to search for. Will apply binary search, so assumes the list is already sorted. Then returns the index of the first occurrence of the item key, or -1 if not found.

GetNumber
This function prompts the user to enter an integer value that will be used to fill up the array elements (array initialization). The inputted value will be validated to be within the range of min..max. The valid entry will be returned. (refer to previous labs – reuse the function!)

InitArray
This function uses the value returned from GetNumber to initialize all array elements with. The input parameters of this function are: name of the array, size of the array and the initialization value. It does not return a value.

PrintArray
This is a void function that accepts the name and size of array as two input parameters and prints the contents of the array.

PopulateArray
This function is void return and it fills array elements with random values in the range of 1 .. MAX/2. It has two input parameters which are the name and size of the array.

SearchArray
This function uses a linear search to look for a certain key (value) in the array elements and returns the number of occurrence(s) for that key among the array elements. It returns zero if no matching value was found.

Sample output of the program is similar to the previous lab.