**Introduction To Computer Science – 150005**

**Homework Assignment #7**

**Searching, Sorting, 2D Arrays**

**Comments:**

1. Every answer must contain a main function that demonstrates the correctness of the functions you have written.
2. Use meaningful variable names.
3. Use functions whenever possible.
4. Comment each function (including a comment before the main function explaining its purpose and how it works). Also, at the end of each program add a comment with a sample run with its output.
5. Be careful on code readability and appearance (indentation)
6. Make sure to compute exactly what is requested in each question.
7. Make sure your answers check correctness of their inputs (including printing ERROR and reenter input.
8. Output of a program is marked in green and input in yellow.
9. Submit the solution according to the directions in Moodle..
10. Remember: submit your own work!

**Definitions:**

**Ascending Order** – the first element is less than the second element and the second element is less than the third element, etc. A list of numbers that is sorted in ascending order does not contain duplicate numbers.

**Non-descending Order** – the first element is less than or equal to the second element and the second element is less than or equal to the third element, etc. Duplicate values can appear in a list of numbers that is sorted in non-descending order.

**Descending Order** – the first element is greater than the second element and the second element is greater than the third element, etc. . A list of numbers that is sorted in descending order does not contain duplicate numbers.

**Non-ascending Order** – the first element is greater than or equal to the second element and the second element is greater than or equal to the third element, etc. Duplicate values can appear in a list of numbers that is sorted in non-descending order.

**Question 1 (binary search)**

1. Write a function that receives via parameters:
   * an array that contains 10 numbers that are stored in ascending order,
   * an integer that represents the size of the array,
   * and an integer that represents the value that the function should search for.

The function should perform a binary search to determine if the number (the value in the third parameter) appears is in the list. If it does, then the function should return the place (index) in the list where the number is found, otherwise it returns -1.

**Reminder:**  the first index of the array is 0.

1. Write a main program that will test the function that was written in part a. The main program should first declare an array of ten integers. The main program should then input a list of 10 ascending integers. If the input is not correct (not in ascending order without duplicates), then the program should print ERROR and input again all ten numbers of the list.  
     
   After successfully inputting the list of ten numbers, the program should input an additional integer after printing enter 1 number:  
     
   The main program should call the function created in part a to see if the last number inputted does indeed appear in the list of ten numbers. If the number is found, then the program prints the number X was found at index Y where X is the number that the function was searching for and Y is its’ index in the list. Otherwise the program should print not found.

Sample runs:

|  |  |
| --- | --- |
| enter 10 numbers:  1 3 5 5 9 11 13 15 17 19  ERROR  1 3 5 2 9 11 13 15 17 19  ERROR  1 3 5 7 9 11 13 15 17 19  enter 1 number:  5  the number 5 was found at index 2 | enter 10 numbers:  1 3 5 7 9 11 13 15 17 19  enter 1 number:  12  not found |

**Question 2 (Merge)**

In class we learned how to merge multiple lists into one. Write a program that reads in the elements of 3 lists. Each list contains at most 10 positive integers (shorter lists are terminated with a 0) and are sorted in strictly descending order (no repeats). The program prints out the merge of the 3 lists.

Note, if the input is not correct (not strictly descending without repeats), the program prints ERROR and inputs again all the integers just for the list that is incorrect.

The merge algorithm should be in a separate function, and there should also be a separate function to read in the inputs (aka there should be **one** function for inputting into an array and that function should be called three times – once for each array inputted/)

Sample run:

enter values for the first vector:

6 4 2 0

enter values for the second vector:

21 19 17 14 13 12 9 6 3 2

enter values for the third vector:

3 2 1 0

merged vector is:

21 19 17 14 13 12 9 6 6 4 3 3 2 2 2 1

Sample run:

enter values for the first vector:

4 2 0

enter values for the second vector:

5 4 3 3 0

ERROR

5 23 0

ERROR

4 3 0

enter values for the third vector:

7 6 5 4 3 2 1 0

merged vector is:

7 6 5 4 4 4 3 3 2 2 1

**Question 3 (InsertionSort)**

In class we learned the insertion sort algorithm. Write a program that inputs 100 positive integers from the userinto a 10\*10 matrix. After inputting the 100 values into the matrix your program should output the message before sorted: and then proceed to output the contents of the matrix such that a space appears between the numbers and each line of the matrix appears on its own line of output.

Your program should then use insertion sort to sort the lower left triangle and the upper right triangle of the matrix in non-descending order.

The sort is performed such that the smallest number in the left triangle appears in the upper left square of the triangle, and all remaining numbers are ordered in the subsequent lines from left-to-right, line after line until the largest number appears in the bottom right square of the triangle. Similarly, in the right triangle the smallest number appears in the upper most left square and al remaining numbers are ordered in the subsequent lines from left to right, line after line until the largest numbers appears in the bottom right.

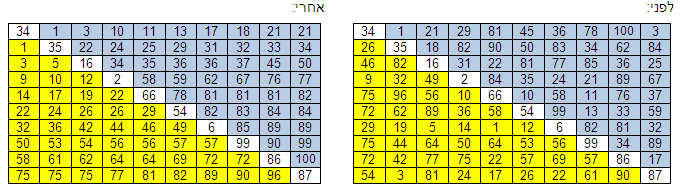
Note, that the diagonal does not belong to either triangle and should remain as is – not sorted.

After the sort is completed, the program should print the message:

sorted matrix:

and then continue to output the contents of the matrix such that a space appears between the numbers and each line of the matrix appears on its own line of output.

Note: You will need to create a sort function.



Sample run:

|  |
| --- |
| enter the values:  1 11 21 31 41 51 61 71 81 91 2 12 22 32 42 52 62 72 82 92 3 13 23 33 43 53 63 73 83 93 4 14 24 34 44 54 64 74 84 94 5 15 25 35 45 55 65 75 85 95 6 16 26 36 46 56 66 76 86 96 7 17 27 37 47 57 67 77 87 97 8 18 28 38 48 58 68 78 88 98 9 19 29 39 49 59 69 79 89 99 10 20 30 40 50 60 70 80 90 100  before sorted:  1 11 21 31 41 51 61 71 81 91  2 12 22 32 42 52 62 72 82 92  3 13 23 33 43 53 63 73 83 93  4 14 24 34 44 54 64 74 84 94  5 15 25 35 45 55 65 75 85 95  6 16 26 36 46 56 66 76 86 96  7 17 27 37 47 57 67 77 87 97  8 18 28 38 48 58 68 78 88 98  9 19 29 39 49 59 69 79 89 99  10 20 30 40 50 60 70 80 90 100  sorted matrix:  1 11 21 22 31 32 33 41 42 43  2 12 44 51 52 53 54 55 61 62  3 4 23 63 64 65 66 71 72 73  5 6 7 34 74 75 76 77 81 82  8 9 10 13 45 83 84 85 86 87  14 15 16 17 18 56 88 91 92 93  19 20 24 25 26 27 67 94 95 96  28 29 30 35 36 37 38 78 97 98  39 40 46 47 48 49 50 57 89 99  58 59 60 68 69 70 79 80 90 100 |

**Question 4 (BubbleSort)**

In class we studied the bubble sort algorithm. Write a program that outputs the message enter 10 words: , inputs 10 words each containing 5 letters, and stores them in a 10 x 5 matrix (i.e., 10 rows, each row containing a single word of 5 characters, one character after the other). The program should use bubble sort to sort the words in lexicographical (dictionary) order.  
After it completes the sort it should output the message

after sorting: and then print out the words. Note, you can assume the input is correct, that is, contains exactly 10 words with 5 letters in each word.   
The sort algorithm should be in a separate function.

Sample run:

enter 10 words:

house apple teach array teach books point float apply begin

after sorting:

apple apply array begin books float house point teach teach