**Introduction To Computer Science – 150005**

**Homework Assignment #7**

**Arrays**

**Comments:**

1. Be careful on code readability and appearance (indentation)
2. Make sure to compute exactly what is requested in each question.
3. In any case of wrong input or error, you must give a message ERROR and allow the user to reenter correct input
4. In each question you must define functions according to need to make the code more readable.
5. Use meaningful variable names
6. Comment each program (including a comment before the main program explaining its purpose and how it works). Also, at the end of each program add a comment with a sample run with its output.
7. Output of a program is marked in green and input in yellow.
8. Submit the solution according to the directions in moodle.
9. Remember: submit your own work! (cannot submit in pairs)
10. Write a program that reads in 15 whole numbers and prints GOOD if the list contains all 15 numbers between 1 and 15 and otherwise NOT GOOD.  
    The input is done using the following message:  
    enter 15 numbers:

Sample run:

|  |  |
| --- | --- |
| enter 15 numbers:  15 8 9 16 1 3 4 2 10 14 6 5 13 12 11  NOT GOOD | enter 15 numbers:  15 8 9 7 1 3 4 2 10 14 6 5 13 12 11  GOOD |

1. Write a program that defines two arrays:
   * Array of integers called *vector1* that contains up to 500 elements
   * Array of integers called *vector2* that contains up to 100 elements

The program inputs:

* the number of elements to input to *vector1*  
   (using the message: enter size of first array:
* corresponding list of numbers to store in *vector1*  
  (using the message: enter first array values: )
* the number of elements to input to *vector2*  
  (using the message: enter size of second array: )
* corresponding list of numbers to store in *vector2*  
  (using the message: enter second array values: )

The program checks and prints out the number of times *vector1* contains the elements in *vector2*, that is, the number of times the elements in vector2 is repeated in vector1.

using the message: result: # times

Sample run:

enter size of first array:

18

enter first array values:

1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6

enter size of second array:

4

enter second array values:

2 3 4 5

result: 3 times

Sample run:

enter size of first array:

11

enter first array values:

1 2 1 2 1 2 1 2 1 2 1

enter size of second array:

3

enter second array values:

1 2 1

result: 5 times

1. Write a program that inputs a list of values into an array of integers. The end of the input is marked with a 0 (which cannot be one of the elements in the list). The maximum number of elements in the array is 100.   
   Use the following message:  
   enter up to 100 values, to stop enter 0:  
   The program should fix the array such that each value appears at most once in the array. Note:
   * The value that remains in the array is the first one that appears in the array
   * Important! To delete all repeating values, the program should pass the array to a void function.

The program should print the remaining elements as well as the number of remaining elements.

The messages used are:  
the new vector is: - for printing the array   
number of elements: # - for printing the number of elements

For example, if the array originally contained 7 15 8 17 8 15 9 14, then the output is: 7 15 8 17 9 14 and the number of elements: 6.

Sample run:

enter up to 100 values, to stop enter 0:

1 5 8 9 5 4 1 2 3 5 1 4 8 9 6 5 0

the new vector is:

1 5 8 9 4 2 3 6

number of elements: 8

1. Write a program that inputs 6 real numbers between 0 and 1 (0 and 1 not included) into an array called *numbers*.   
   You can assume the numbers are all different (no need to check  
     
   The program should create a new array called *indices* of integers, which contains the indices of the numbers in the array *numbers* in sorted ascending order. That is, the first element in *indices* contains the index of the smallest number in *numbers*, the second element in *indices* is the index of the second smallest number in *numbers*, and so on.  
   For example, if *numbers* contains = {0.5, 0.95, 0.1, 0.7, 0.19, 0.86}, then the first element in *indices* would be 2 since *numbers*[2]=0.1 is the smallest number in *numbers*. The second element in *indices* is 4 since *numbers*[2]=0.19 is the second smallest number.  
     
   Note: If the input is incorrect (contains a number that is not between 0 and 1), print the message ERROR and input all 6 numbers again.

Sample run:

enter 6 numbers between 0 and 1:

0.3 0.2 0.4 0.1 0.05 0.9

sorted indices:

1 2 4 5 3 0

Sample run:

enter 6 numbers between 0 and 1:

0.9 0.5 2.70.3 0.2 0.4 0.1 0.05 0.9

sorted indices:

ERROR

1 2 4 5 3 0

1. The set difference between two sets set1 and set2 is the set that contains all elements in set1 that are **not** in set2.   
   For example:  
   if set1= {1,3,5,7,9}   
   and set2={1,2,3,4,5},  
   then the set difference = set1- set2 = {7,9}.  
   Write a program that reads in 6 positive integers into the array set1 and 6 positive integers into the array set2 and creates a third array called difference that contains the set difference of set1 and set2.  
   If difference is not empty, the program prints out its elements. If difference is empty, then the program prints empty.  
   You can assume that there are no repeating elements in either set, but the sets are not necessarily ordered.  
   The sample run demonstrates the messages you would write for the user to input the values. Use the exact messages as noted in the example.

Sample run:

enter first 6 numbers:

1 3 5 7 9 10

enter next 6 numbers:

1 2 3 4 5 6

set difference is:

10 9 7

Sample run:

enter first 6 numbers:

1 3 5 7 9 10

enter next 6 numbers:

1 3 5 7 9 10

set difference is:

empty