**Array Problems**

You are expected to write your algorithms as black boxes, i.e. procedures.

1. Assume that you have a single dimensional array called *numbers* that hold 10 integer values. Write an algorithm in ADL that searches a given integer in *numbers* from keyboard. You need to print "found" if the item is found. Otherwise, you will print "not found".

declare numbers[1..10]

1. Assume that you have a single dimensional array called numbers that hold 10 integer values. Write an algorithm in ADL that sorts the values in the array in ascending order.

declare numbers[1..10]

1. Assume that you have a single dimensional array called numbers that hold 10 integer values. Write an algorithm in ADL that sorts the values in the array in descending order.

declare numbers[1..10]

1. Assume that you have a single dimensional array called numbers that hold 5 integer values. Write an algorithm in ADL that reverses the values held in the array.

For example,

declare numbers[1..5]

1

5

3

2

7

The inverse of this array

7

2

3

5

1

would look like this:

1. Assume that you have a single dimensional array called numbers that hold 10 sorted integer values in ascending order. Write two algorithms in ADL that search a given integer from keyboard in *numbers*. You need to print "found" if the item is found. Otherwise, you will print "not found".

1

2

3

5

7

declare numbers[1..5]

1. Assume that you have a single dimensional array called *string* that holds 200 characters (i.e. spaces, digits, letters and punctuation characters). Write an algorithm in ADL that counts the number of words in *string*. Note that words do not have to start from the beginning, e.g. a paragraph.

declare string[1..200]

H

i

,

a

r

e

y

o

u

For example, in the above string, there are three words. In order to write this algorithm, you can rely on the following three abstractions:

procedure eatSpaces(IN string[], INOUT index)

procedure eatCharacters(IN string[], INOUT index)

procedure isAlphaNumeric(IN character, OUT result)

The eatspaces procedure takes an array and its current index, and updates the index, so that it points to a character value.

The eatCharacters procedure takes an array and its current index, and updates the index, so that it points to a space.

The isAlphaNumeric procedure takes a character and determines if it is a letter or a digit (i.e. alphanumeric). For example, "a" and "1" are both alphanumeric characters.

1. Assume that you have a single dimensional array called *string* that holds 200 characters (i.e. spaces, digits, letters and punctuation characters). Write an algorithm in ADL that enters a series of characters that form a word and searches this word in *string*. You need to print "found" if the item is found. Otherwise, you will print "not found".

declare string[1..200]

1. Assume that you have two single dimensional arrays called *string1* and *string2* that hold 200 characters each. Write an algorithm in ADL that concatenates both strings. The concatenated string will be held in an array called *longString* that holds 400 values.

declare string1[1..200]

declare string2[1..200]

declare longString[1..400]

1. Assume that you have a single dimensional array called *set* that holds 10 integer values. Write an algorithm in ADL that determines if *set* is a mathematical set. Note that a mathematical set has two properties. First, the order of elements in a set is immaterial. Second, no elements in a set must be repeated. We cannot write an algorithm to determine the first property. Hence, for the purpose of this algorithm determining the second property will be sufficient to determine whether or not *set* is a mathematical set. For more information on sets refer to [this link](http://www.mathsisfun.com/).
2. Assume that you have two single dimensional arrays called *set1* and *set2*, each of whichholds 10 integer values. Write separate algorithms in ADL to calculate the following:
3. the intersection of the two sets. The resulting set should be called *intersection*.
4. the union of the two sets. The resulting set should be called *union*.
5. the difference of the two sets. The resulting set should be called *difference*.