Problem Set 3 Exercise #12: Set Containment

Reference: Lecture 7 notes

Learning objectives: One-dimensional array; Algorithm design

Estimated completion time: 45 minutes

Problem statement:

Write a program **set_containment.c** to read two arrays **`arrA`** and **`arrB`** of distinct **`int`** values (at most 10 each), check whether numbers in **`arrA`** is a subset of numbers in **`arrB`**.

For example, array {1, 3, 5} is a subset of another array {1, 5, 3, 9}.

Your program should contain a function

```
    void scan_array(int arr[], int size)
```

that reads elements into array **`arr`**, and another function

```
    int is_subset(int arrA[], int sizeA, int arrB[], int sizeB)
```

to check if numbers in **`arrA`** is a subset of numbers in **`arrB`**. It returns 1 if so, 0 otherwise.

You may assume that 1 ≤ **`sizeA`** ≤ 9 and 1 ≤ **`sizeB`** ≤ 9.

Note:

This task requires both universal (for all) and existential (there exists) argument, and involves nested loops. Please think about your algorithm carefully before coding. A flawed/complicated algorithm makes your coding and debugging much tougher.

Sample run #1:

```
Size of 1st array? 4
        Enter 4 values: 14 5 1 9
Size of 2nd array? 7
        Enter 7 values: 2 9 3 14 5 6 1
1st array is a subset of 2nd array
```

Sample run #2:

```
Size of 1st array? 4
        Enter 4 values: 14 5 1 9
Size of 2nd array? 6
        Enter 6 values: 2 3 14 5 6 1
1st array is not a subset of 2nd array
```