Handy Array Library – searching

Work to do

Let's extend the work started in 201-A01 and 201-A02 by adding a searching function. One of the perks of having a sorted array is that we can efficiently search for a value inside of it using a binary search algorithm.

- **int intarray_is_sorted (const int data[], int size);**
  This function tests if the integer array **data** is sorted in ascending order and returns 1 if it is or 0 if it is not.
- **int intarray_search (const int val, const int data[], int size);**
  This function search for the value **val** in the sorted integer array **data** and return 1 if it is found and 0 otherwise. You will start by ensuring the array is sorted in ascending order by calling **intarray_is_sorted**. If the integer array **data** is not sorted, return -1 as a sign that something was not appropriate in the parameters.

You will try to come up with the binary search algorithm on your own. Do not refer back to the text, the objective of this exercise is to force you to come up with your own solution, possibly using tidbits of information you remember from your readings

Example(s)

n/a

Hints

- n/a

Testing

Add to the main function in tests.c some code to search in the sorted the array for values specified by the user. You will keep asking the user for values to search and each time display the whole array as well as whether the specified value was found in it or not.

Once this part of the test is done, ask the user to input another array, do not sort it, and try to call the search function on it with a user specified value.

This will allow you to test whether your function return -1 as it should in these cases.

Input	Output	
Input array	Expected	Observed