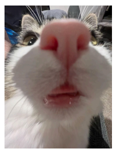

Assignment 5: Sorting & Searching

Vaani Goenka
15th June 2025
Collaborators: None
Formally verified? Yes

Problem 1: Searching - Everyone's a suspect!

BLUEPRINT

Requires:
A static array of type int and a target search element also of type int.

Ensures:
Every element is compared to the target element, and the output of the program is the index of the target integer in the array, or -1 if it does not exist in the given array.

OPERATIONAL STEPS

Step 1:
We want to store the index of the element under consideration while iterating and the index of the found element. We keep two variables for them. We initialise the latter with -1 such that if no element is found, then the default value is returned.

Step 2:
We want to look at every element one by one. We use a loop over the entire array. To know the bounds of the array, we need the length of the array to set that as the upper limit. While the index is less than the length of the array, we

Step 3:
Consider the ith element of the array, where i = current index. If this = target, then we set index = i, and set i = len so that the loop breaks immediately.

Step 4:
Else, we increment i. The loop runs again provided the loop condition is valid, and the next element is compared and so on...

OCAML CODE

```

                    (* getLen and getElem are user-defined functions. *)

                    let search (arr : int list) (target : int) : int =
                        let i = ref 0 in (* Step 1 *)
                        let index = ref (-1) in (* Step 1 *)
                        let len = List.length arr in
                        while !i < len do  (* Step 2 *)
                            match List.nth arr !i with
                            | n when n=target -> index := !i; i := len; (* Step 3 *)
                            | _ -> i :=!i + 1; (* Step 4 *)
                        done;

                    !index
                ;;
```

PROOF

INVARIANT

Invariant Condition:

After any kth iteration, if index = -1 then the element does not exist in arr[0 : k], (k inclusive), else if index = k then the element has been found at index.

0 <= k <=length arr - 1

0 <= index <=length arr - 1

Pre-condition:

Before the loop begins:

index = -1 from *line 6* and no element has been compared yet -> element has not been found yet

Invariant holds.

After the ith iteration:

Assume the invariant holds after the ith iteration.

After the (i+1)th iteration:

If the ith element = target, then index = i, i = len and the loop breaks before the (i+1)th iteration from *line 10*. Therefore, the invariant condition holds as target has been found at index = i.
Else, it continues to check the (i+1)th element from *line 11*, and if (i+1)th element = target then index = i+1 and the loop breaks. Therefore, the invariant condition holds. Else, we know from the ith step that the element does not exists in arr[0 : i], and it is not the (i+1)th element either, so index remains -1. The invariant condition holds.

Post-condition:

i >= length arr, so the loop breaks from *line 8*. If index = -1, then the element does not exist in arr[0 : length arr - 1] which is true. The only other possibility is that 0 <= index <=length arr - 1, denoting that the target exists at index (found at i). Therefore, the invariant condition holds. The correct value is returned from *line 14*