[19sp]:

* (Placeholder for todos added from Slack)

(The slack bot didn’t work)

We need to add a few tests for rainfall. We should test a case like [-1, -999] as I had a few students only return -1 in the case of an empty list or in the case of [-999;...] and they passed all tests

~~--------------------------------------------------------------------------------------------------------------------------------~~

Everything below resolved:

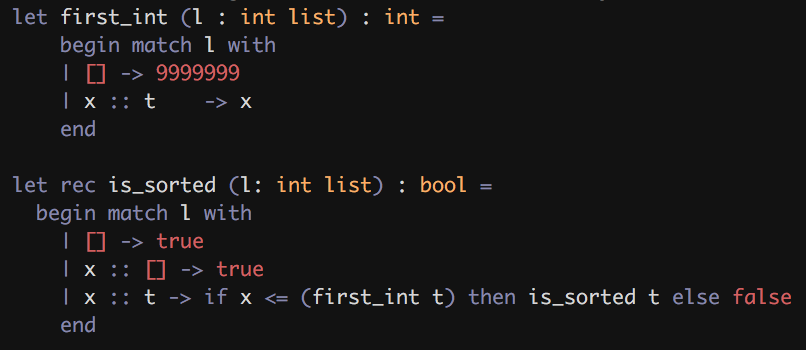
~~MC [16sp]: add tests to move\_robot that start at 99 and move backwards or starts at 0 and move forward. This code passed all of our tests:~~

~~let rec move\_robot … : int =~~

~~if pos = 0 || pos = 99 || num\_moves = 0 then pos else …~~

~~MC [16fa]: same ^^~~

~~JH [16fa]: Add tests to prevent this…~~



---- 17fa ---

~~[PP]: hw1: better test cases for exists and sublist on the grading server~~

~~I’ve seen a handful of wrong implementations that have received full credit for both functions~~

~~For exists, I would have a case where the true is really far into the list, like `exists [false;false;false;false;false;false;false;false;false;true]` so that they can’t brute force it.~~

~~For sublist, I would put more complicated examples of subsequences that aren’t sublists.~~

--- 18sp ---

~~[PP]: hw1: rainfall: Add a test with a list of only zeros. Many students return -1 here when they should return 0. sublist: Add harder test cases, like `sublist [1; 2; 3; 4] [1; 2; 0; 0; 0; 3; 4]`, so that they can’t special case it for subsequences that only have one element breaking contiguity.~~

~~[ES] following works for sublist:~~

~~let rec sublist (l1: int list) (l2: int list) : bool =~~

~~begin match (l1, l2) with~~

~~| ([], \_) -> true~~

~~| (\_, []) -> false~~

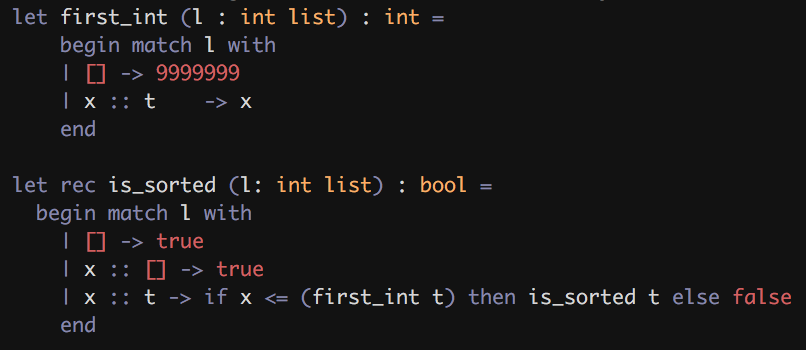
~~| (list1, head :: tail) -> if list1 = head :: tail then true else sublist~~

~~list1 tail~~

~~end~~

~~→~~ **~~Solution: Add test for sublist [1;2] [1;2;3] so it does not pass~~**

---- various ----

* ~~Remove comments about “printing” as a great debugging technique~~
* ~~HW1 - fix tests with = true or = false (i.e. don’t do <bool> = true, just <bool>. And don’t do <bool> = false, just not <bool>)~~
* ~~People get confused about not(...) in the test cases (explain it, don’t just get rid of it)~~
* ~~HW 1: add tests to move\_robot that start at 99 and move backwards or starts at 0 and move forward. This code passed all of our tests:  
  let rec move\_robot … : int = if pos = 0 || pos = 99 || num\_moves = 0 then pos else …~~
* 

~~- An idea: ask them to turn in test cases 1 day before the assignment is due. Then run the test cases on both successful and unsuccessful assignments and make sure that they catch bugs. Perhaps for early assignments, give them as many submissions as they need to catch n bugs in a sample solution?~~

~~-- Assign a committee to read student feedback and report back~~

FROM REPO (**very old as of 18sp**):

Students don't realize that they shouldn't comment out or delete functions

that we have in the interface that they have not implemented (especially for

challenge problems). We should make this more clear. -PM

[ES] The comments above might make the homework too commented

~~When explaining the concatenation operator, we have a line that says~~

~~;; if "a" ^ "b" <> "ab" then failwith "impossible"~~

~~this is actually pretty confusing and could be explained a bit better. -PM~~

~~(The above line should be removed from the files. -SZ)~~

~~The very first question about Tax preparation raised a lot of issues:~~

~~- floating point precision (rounding)~~

~~- overflow~~

~~- values less than 100000~~

~~I recommend that we replace it with something better. -SZ~~

Overall the assignment was good. Notes for next year:

- Another issue is with printing/toplevel. It would be good if the students could actually do the assignment without using the toplevel. To do that, they need to know how to print out lists (and other things.) But, printing lists is a fine recursive function to cover, both in class and in the homework assignment.

[ES] printing out lists is already a problem (join / int\_join) and printing out trees / queues is weird, just leave as tests

~~- There is a lot of confusion between "::" and "@" for beginning OCamllers. Maybe move append into the lecture or lab to give it more prominence.~~

- An idea: ask them to turn in test cases 1 day before the assignment is due. Then run the test cases on both successful and unsuccessful assignments and make sure that they catch bugs. Perhaps for early assignments, give them as many submissions as they need to catch n bugs in a sample solution?

[ES] maybe a good idea for other courses? We’re good