Last name	
First name	
Group	
Tutorial teacher	

Grade

Algorithmics Midterm Exam 1 - Part 2

Undergraduate 1^{st} year S1

Epita

9 Nov. 2020 - 8:30

$\hfill\square$ This is the part 2 of the subject - You have to give back the two parts!

 \Box You must answer on this subject.

- Answer within the provided space. Answers outside will not be marked.
- Penciled answers will not be marked.

 \square CAML :

- All CAML code not indented will not be marked.
- In the absence of any indication in the document, the only functions that you can use are failwith and invalid_arg (no other predefined function of CAML).
- Any CAML code must be followed by the result of its evaluation: the CAML answer.

 $\hfill\square$ The presentation is marked.

Exercise 2 (Deletion – 4 points)

Write the function delete x list that removes the first appearance of the value x (if it is present) from the sorted (in increasing order) list list.

Exercise 3 (Insertion at the rank i - 5 points)

Write the function insert_nth x i list that inserts the value x at the rank i in the list list. The function has to raise an exception Invalid_argument if i is negative or zero, an exception Failure if the list is too short.

Application examples:

insert_nth 0 5 [1; 2; 3; 4; 5; 6; 7; 8; 9];; - : int list = [1; 2; 3; 4; 0; 5; 6; 7; 8; 9] # insert_nth 0 10 [1; 2; 3; 4; 5; 6; 7; 8; 9];; - : int list = [1; 2; 3; 4; 5; 6; 7; 8; 9; 0] # insert_nth 0 12 [1; 2; 3; 4; 5; 6; 7; 8; 9];; Exception: Failure "out of bound". # insert_nth 0 (-2) [1; 2; 3; 4; 5; 6; 7; 8; 9];; Exception: Invalid_arg "negative rank".



Exercise 4 (Double Search – 4 points)

Write the function search_both $list \ a \ b$ that tests whether the two distinct values a and b are in the list list.

 $Application \ examples:$

```
# search_both [12; 5; -4; 0; 7; 21; 3] 5 0;;
- : bool = true
# search_both [12; 5; -4; 0; 7; 21; 3] 21 (-4);;
- : bool = true
# search_both [12; 5; -4; 0; 7; 21; 3] 0 42;;
- : bool = false
```



Exercise 5 (Mystery – 2 points)

The go function is defined as

```
let go = function
   [] -> []
   | e::list ->
   let rec what x = function
        [] -> []
        | e::list -> (e * x)::(what e list)
   in
   what e list;;
```

Give the results of the successive evaluations of the following phrases.

go [1; 1; 1; 1; 1] ;;

♯ go [42] ;;

go [1; 2; 3; 4; 5] ;;

go [2; 21; 2; 21; 2; 21] ;;

