Don’t forget to Register!!

Connect to the web ML interpreter at http://try.ocamlpro.com/ You may also want to open a simple editor (eg. WordPad) where you can write your programs. It is sufficient to cut and paste the programs in the provided window on the web page and the ML interpreter will acquire them. You may want to refer to lecture slides for examples of the accepted syntax. Please remember that the return key will cause the evaluation of the current definition, while the shift + return keys will allow you to enter multi-line definitions (use shift + return at the end of the line to move to a new line). You can use up and down arrow to cycle across the given commands.

1. Simple programs

Consider the following two definitions (you should be able to cut&paste them in the interpreter's window. It is possible that the pasted text does not display properly, until you press return):

```ocaml
let double n = n + n
let square n = n * n
let avg (n, m) = (n + m) / 2
```

a. Compute the double and the square of 2?

b. What is the average of the two results obtained above?

c. What is the type of the function `avg`? Can you explain it?

d. Which error do you get if you try `avg 4 4`? Can you explain it? Hint: refer to c. above.

e. Have you obtained the average by using the numerical results of point a.? Compute the average by composing the provided functions.

f. Are the above definitions satisfactory? (if no go to h.)

g. Try to compute the average of 1 and 2. Is the result correct?

h. The function `avg` needs improvements. Could any issues have been spotted by the answer in c.? Please amend the definition of `avg`. Hint: you can find a clue in lecture slides.

i. Check that the average in g. is properly computed (you may need to slightly adjust the input).

j. Extra: Does your upgraded definition work also when you re-compute an average by composing functions, as did in g. ?
2. Pattern matching and recursion

Using recursion and pattern matching as appropriate solve the following problems.

a. Define a recursive function that sums all the elements in a list of integer.

b. Define a recursive function that sums all the elements in a list of integers excluding the last item.

3. Higher Order Functions

Choose one of the functions defined above that can be applied to an integer.

a. Define a recursive function that applies a given function to each element of a list of integers. Try your definition by using the function chosen above.

b. Define a recursive function that applies twice a given function to each element of a list of integers (that is the function is applied to the element and then again to the result obtained, for each element of the list). Try your definition by using the function chosen above.

Checkpoint

Now demonstrate to a tutor that you have successfully solved the problems in 1. and in either 2. or 3. (or both!). Then, attempt point 4.

4. Polymorphism (extra)

Using recursion and pattern matching, define a recursive function that reverses a list, e.g. given [1;2;3] it returns [3;2;1]. You may want to use the following function `merge` that merges two lists.

Hint: reasoning recursively may help here: if you know how to reverse the tail of the list ..... Also, you may need to be careful about the distinction between elements and lists, e.g. `merge` works with lists.

```plaintext
let rec merge(r, s) = match r with
| [] -> s
| h::t -> h :: merge(t, s)
```