Recitation 2 : Introduction to OCaml

1. Write down the type of the following in OCaml:
   a) `let double x = 2*x;;`
   b) `let square x = x*x;;`
   c) `let twice f x = f (f x);;`
   d) `let quad = twice double;;`
   e) `let fourth = twice square;;`

2. Write down the types of the following in OCaml:
   a) `let tripleFloat x = 3.0*.x;;`
   b) `let thrice f x = f(f(f(x)));;`
   c) `let composition f g x = f(g(x));;`
   d) `let div x y = x/y;;`
   e) `let triple3 = thrice tripleFloat;;`

3. Generalize `twice` to a function `repeat`, such that `repeat f n x` applies `f` to `x` a total of `n` times. That is,
   - `repeat f 0 x` yields `x`
   - `repeat f 1 x` yields `f x`
   - `repeat f 2 x` yields `f (f x)` (which is the same as twice f x)
   - `repeat f 3 x` yields `f (f (f x))`

4. What is the type of the following function? What will its output be for the inputs (i) `1`, and (ii) `["a"; "b"; "c"; "d"]`?

```
let f list =
  let rec aux acc = function
    | [] -> acc
    | h::t -> aux (h::acc) t in aux [] list;;
```

5. Write a function to remove the nth element from a list.

6. Write an OCaml function to return the last element of a list. THEN, find out whether OCaml offers a built-in function to do this for you.

7. One way of showing how Lambda calculus is "used" in a language like OCaml is the let e1 in e2 structure. For example, let x = 5 in let y = 3 in x+y;; is just like (λx.λy.(x+y)) (5) (3) in lambda calculus. Try making similar conversions with the following OCaml code statements:
   a) `let x = 4 in let y = 12 in y/x;;`
   b) `let x = 3 in let y = 10 in let z = 5 in (x*y)/z;;`
   c) `let f x = x + 3 in let y = 5 in f y;;`