```
Last login: Thu Mar 21 12:49:34 on ttys000
ex-remote:~ marcob$ ocaml
        OCaml version 4.02.2


(* una prima espressione *)
# 3+4*2;;
- : int = 11
(* R: Top Evaluation Level: anonimo : tipo int = pres *)
#
# let x = 5;;
val x : int = 5
(* R: Top Evaluation Level: val x : tipo int = presentazione *)
#
# x;;
- : int = 5
(* R: Top Evaluation Level: anonimo : tipo int = presentazione *)
#
#
( * booleani *)
# true && (x>5);;
- : bool = false
#
#
(* liste *)
# [];;
- : 'a list = []
# "x"::[];;
- : string list = ["x"]
# x::[];;
- : int list = [5]
#
#
(* Una funzione: Solo funzioni monadiche *)
# fun x -> fun y -> x + y;;
- : int -> int -> int = <fun>
(* R: Top Evaluation Level: anonimo : tipo int = presentazione *)
#
# let tuple = fun x -> fun y -> (x,y);;
val tuple : 'a -> 'b -> 'a * 'b = <fun>
#
# let add =  fun x -> fun y -> x + y;;
val add : int -> int -> int = <fun>
# let sum = fun (x,y) -> x + y;;
val sum : int * int -> int = <fun>
(* applichiamole *)
# (add 3 7)*sum(3,7);;
- : int = 100
#
(* una funzione ricorsiva su liste *)
#  let rec nTok = fun n -> fun k -> if k>0 then n::nTok(n+1)(k-1) else [];;
val nTok : int -> int -> int list = <fun>
(* applichiamola *)
# nTok 2 7;;
- : int list = [2; 3; 4; 5; 6; 7; 8]
#
#
(* Tipi Algebrici *)
# type stree = Empty | Root of string * stree list;;
type stree = Empty | Root of string * stree list
# let mkE = fun () -> Empty;;
```

```
val mkE : unit -> stree = <fun>
# let mkS = fun root -> fun sons -> Root(root,sons);;
val mkS : string -> stree list -> stree = <fun>
(* applichiamole *)
# let t1 = mkS "a" [mkS "t" [];(mkS "c" [mkS "r" []])];;
val t1 : stree = Root ("a", [Root ("t", []); Root ("c", [Root ("r", [])])])
# let t2 = mkS "e" [];;
val t2 : stree = Root ("e", [])
# let t3 = mkS "d" [mkS "e" []];;
val t3 : stree = Root ("d", [Root ("e", [])])
# let t = mkS "b" [t1;t2;t3];;
val t : stree =
  Root ("b",
   [Root ("a", [Root ("t", []); Root ("c", [Root ("r", [])])]);
    Root ("e", []); Root ("d", [Root ("e", [])])])
(* Esercizio
    Definire una funzione OCaml che calcoli la frontiera di un albero di tipo stree.
*)
```