OCAMEL

Regarding physical and structural equality: characters, integers, unit and empty list can be equal both structurally and physically. I.e. let i = 3 and j = 3;; both i == j and i = j will return true both i != j and i<> j will return false However, the case is different for lists for example: let i = [1;2] and j = [1;2];; i = j will return true (because contents are the same) BUT i == j will return false (because i and j are different lists) Same thing for <> (will return false) and != (will return true) HTH

= Checking structural equality

== checking physical equality

Pattern matching another way

# let rec switch i =

  match i with

  1 -> 1

  | \_ -> i \* switch (i -1);;

val switch : int -> int = <fun>

or

let rec another = function

  1 -> 1

  | n -> n \* s(n -1);;

MUTUALLY RECURSSIVE FUNCTION

can call with any using any function

let rec fun1 x =

  match x with

  0 -> "Zero"

  |\_ -> fun2 x

  and

  fun2 y =

  match y with

  1-> "One"

  |\_->fun1 (y-1)

  ;;

val fun1 : int -> string = <fun>

val fun2 : int -> string = <fun>

LIST

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

Must be same type

MUST not change the element once add in it, IMMUTABLE variable

list can have duplicate of the elements

can define list

let x = [‘a’;’b’]

INSERT to list ‘c’::x;; -> insert to the front [‘c’;’a’;’b’]

TO APPEND -> x@y

List.hd x;; //first element of list x

List.tl x;; //all elements of list x except first elements

has\_element in list

1.

let rec has\_element l e =

match l with

[] -> false

|h::t -> if h = e then true

else

has\_element t e;;

DUPLICATE

let rec duplicate ls =

  match ls with

  [] -> []

  |h::t -> h::h::duplicate t;;

val duplicate : 'a list -> 'a list = <fun>

reverse the list

  let rec reverse\_list ls rls =

  match ls with

  [] -> rls

  | h::t -> reverse\_list t rls@[h];;

val reverse\_list : 'a list -> 'a list -> 'a list = <fun>

# reverse\_list["a";"b";"c";"d"] [];;

- : string list = ["d"; "c"; "b"; "a"]