Problem 1 (10 pts): Write a function called `even_indices` which takes any type of list and returns a list of elements at even indices 0, 2, 4, etc. Example uses from a REPL are shown. *Hint: a recursive solution which “skips” element is effective. My if/else solution is 13 lines long while pattern matching makes this considerably shorter.*

```ocaml
# #use "even_indices.ml";;
val even_indices : 'a list -> 'a list = <fun>
# even_indices [];;
- : 'a list = []
# even_indices [0];;
- : int list = [0]
# even_indices [0; 1];;
- : int list = [0]
# even_indices [0; 1; 2; 3; 4; 5];;
- : int list = [0; 2; 4]
# even_indices [0; 1; 2; 3; 4; 5; 6; 7; 8];;
- : int list = [0; 2; 4; 6; 8]
# even_indices ["a"; "b"; "c"; "d"];;;
- : string list = ["a"; "c"]
```

Problem 2 (10 pts): Source code for the `array_fill` function is provided along with a short session which attempts to demonstrate the function. A warning is given on loading the code and an unexpected result occurs. Describe the following.

(A) Why is the warning given?

(B) Why is the array apparently unchanged?

(C) How can the function be corrected to remove the warning and carry out its intended purpose?

```ocaml
> cat -n fill.ml
1 (* fill array with given element *)
2 let fill_array arr elem =
3 for i=0 to (Array.length arr)-1 do
4   arr.(i) = elem;
5 done;
6;;

> ocaml
# #use "fill.ml";;
File "fill.ml", line 3, characters 4-18:
Warning 10: this expression should have type unit.
val fill_array : 'a array -> 'a -> unit = <fun>

# let a = [|9;5;2|];;
val a : int array = [|9; 5; 2|]

# fill_array a 7;;
- : unit = ()

# a;;
- : int array = [|9; 5; 2|]
```
Problem 3 (10 pts): Complete the pointer diagram to shown to reflect how the OCaml code will use existing cons boxes and create new ones.

```
let listX = [6; 1; 2];;

let ansA = List.hd (List.tl listX);;

let ansB = List.tl (List.tl listX);;

let ansC = 7 :: listX;;

let ansD = 9 :: 5 :: ansB;;
```

Problem 4 (10 pts): Write a function called `firstlast` which returns a list of the first and last elements of a parameter list. For empty lists, the empty list is returned. For single element lists, only that element is returned. For full credit, make use of a tail-recursive helper function to complete the function.

Write your code for `firstlast` here.

```ocaml
(* REPL demo for firstlast *)
# firstlast [];;
- : 'a list = []
# firstlast ["a"];;
- : string list = ["a"]
# firstlast ["a"; "b"];;
- : string list = ["a"; "b"]
# firstlast ["a"; "b"; "c"; "d"];;
- : string list = ["a"; "d"]
# firstlast ["a"; "b"; "c"; "d"; "e"; "f"];;
- : string list = ["a"; "f"]
# firstlast [1;2;3;4;5;6];;
- : int list = [1; 6]
```