```
type dir = {
  mutable name : string;
  mutable subdirs : dir list;
  up : dir
  }
exception Path_Error;;
let rec root = {name="/"; subdirs = []; up = root};;

let working_dir = ref root;;

(*Q3*)

(*Q4*)
let mkdir d = let dir = {name = d; subdirs = []; up = !working_dir}
              in
              (!working_dir).subdirs <- dir :: (!working_dir).subdirs;;

type path = string list;;

(*Q5*)
let path_to r =
  let rec path r =
    if r.up == r then [r.name]
    else List.append (path r.up) [r.name]
  in
  let a : path = path r
  in
  a;;

(*Q6*)
let pwd () = let chemin =
               let path = path_to !working_dir
               in
                 let rec to_string p =
                   match p with
                   | [] -> ""
                   | h::d -> h ^ (to_string d)
                 in to_string path
             in print_string chemin;;


(*Q7*)
let lsR () =
  let rec ls dir n =
    let rec print_space n =
      if n > 0 then begin print_string "  "; print_space (n-1) end
    in print_space n;
    Printf.printf "%s\n" dir.name;
    List.iter (fun a -> ls a (n+1)) dir.subdirs
  in ls (!working_dir) 0;;


(*Q8*)
let rec find nom l =
  match l with
  | [] -> raise Not_found
  | h::d -> if h.name = nom then h
            else find nom d;;

(*Q9*)
let go_to p =
  let rec go dir p =
```

```
63    match p with
64    | [] -> dir
65    | h::d -> try let subdir = find h dir.subdirs in go subdir d with  Not_found
66                -> raise Path_Error
67  in
68  let tete = List.hd p
69  in
70    if tete = "/" then
71        go root (List.tl p)
72    else
73        go !working_dir p
74
75(*Q10*)
76let cd p =
77  try
78    let dir = go_to p in
79    working_dir := dir
80  with Path_Error
81  ->raise Path_Error
```