type 'a qnode = { v: 'a; mutable next: 'a qnode option }

let deq (q: 'a queue) : 'a =
begin match q.head with
| None -> failwith "error: empty queue"
| Some qn ->
begin match q.tail with
  | Some qn2 ->
    if qn == qn2 then (q.head <- None; q.tail <- None; qn2.v)
    else (q.head <- qn.next; qn.v)
  | None -> failwith "invariant violation"
end
end

let to_list (q: 'a queue) : 'a list =
let rec loop (qn: 'a qnode option) (acc: 'a list) : 'a list =
begin match qn with
  | None -> List.rev acc
  | Some qn1 -> loop qn1.next (qn1.v :: acc)
end
in loop q.head []
Consider the following code segment.

```ocaml
let q = { head = None; tail = None } in
let n1 = { v = 3; next = None } in
let n2 = { v = 4; next = Some n1 } in
let n3 = { v = 5; next = Some n2 } in
q.head <- Some n3;
q.tail <- Some n1;
n1.next <- Some n3;
let x = deq q in
(* ... *)
```

1. Draw the state of the ASM after the execution of the code segment. You do not need to include the workspace.

![ASM state diagram](image)

2. What is the result of calling `to_list` on `q`?

3. Write the type of each of the following OCaml expressions in the blank provided, or `ill-typed` if the expression does not type check.

(a) let a : ____________________________ = deq

(b) let b : ____________________________ = n1.next

(c) let c : ____________________________ = n3.next.next

(d) let d : ____________________________ = n1.next <- Some n3