This question has been translated from Standard ML to OCaml

(a) Write brief notes on reference types in OCaml and on control structures for imperative programming. [6 marks]

Consider the following OCaml type:

```ocaml
type 'a meal = Snack of 'a
    | Lunch of 'a meal * 'a meal
    | Feast of 'a meal * 'a meal * 'a meal
```

(a) Write a function that is equivalent to `snacker` below but makes no use of references. Briefly explain why the two functions are equivalent.

```ocaml
let snacker m =
    let l = ref [] in
    let munch = function
        | Snack x -> (l := x :: !l)
        | Lunch (m1, m2) -> (munch m1; munch m2)
        | Feast (m1, m2, m3) -> (munch m1; munch m2; munch m3)
    in
    munch m; !l
```

[b] Write a function `gluttony` such that `gluttony m1 m2` makes a copy of `m1`, replacing every `Snack` node with `m2`. [3 marks]

(c) Write a function `glut` such that `glut k m1 m2` makes a copy of `m1`, replacing the `k`th `Snack` node with `m2`. Nodes are counted from left to right, with the leftmost node being number one. [6 marks]