This question has been translated from Standard ML to OCaml

(a) Write brief notes on fun-notation and curried functions in OCaml. Illustrate your answer by presenting the code for a polymorphic curried function replicate, which given a non-negative integer \( n \) and a value \( x \), returns the list \([x;\ldots;x]_n\). [6 marks]

(b) Write brief notes on references in OCaml. Illustrate your answer by discussing (with the aid of a diagram) the effect of the following two top-level declarations:

\[
\text{let rlist = replicate 4 (ref 0) @ List.map ref [1; 2; 3; 4]}
\]
\[
\text{let slist = List.map (fun r -> ref !r) rlist}
\]

[6 marks]

(c) The following three lines are typed at the OCaml top-level, one after the other. What value is returned in each case? Justify your answer clearly. [Note: Recall that an expression of the form \( v := E \) has type \text{unit}.]

\[
\text{List.map (fun r -> (r := !r + 1)) rlist}
\]
\[
\text{List.map (fun r -> (r := !r - 1; !r)) rlist}
\]
\[
\text{List.map (fun r -> (r := !r + 3; !r)) slist}
\]

[8 marks]