A programming language has expressions $e$ with the following syntax:

$$e ::= x \mid n \mid e + e' \mid e(e') \mid (e) \mid \text{let } x = e \text{ in } e' \mid \text{letsta } f(x) = e \text{ in } e' \mid \text{letdyn } f(x) = e \text{ in } e'$$

where $f$ and $x$ range over identifiers and $n$ ranges over numbers. The three let variants introduce simple variables (let) and (non-recursive) functions whose variables are statically (letsta) or dynamically (letdyn) bound.

Using $e$ itself (or any related language whose relationship to $e$ is explained) as abstract syntax define an evaluator $eval$ which, when given an expression $e$ and an environment $\rho$, yields the value of evaluating $e$ in $\rho$. The evaluator can be written in a language of your choice or in mathematical pseudo-code. [12 marks]

Explain carefully in one sentence each:

(a) the forms of value which $eval$ may return;

(b) the form(s) of value which constitute the environment;

(c) the use(s) of environment(s) in letsta and in a call to a function defined by letsta;

(d) the use(s) of environment(s) in letdyn and in a call to a function defined by letdyn. [8 marks]

Hint: because both letsta and letdyn functions may be applied using the same function call syntax, you may find it helpful to use separate forms of value for the two forms of functions.