Compiler Construction

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 $\text{let}$ variants introduce simple variables ($\text{let}$) and (non-recursive) functions whose variables are statically ($\text{letsta}$) or dynamically ($\text{letdyn}$) bound.

Using $e$ itself (or any related language whose relationship to $e$ is explained) as abstract syntax define an evaluator $\text{eval}$ 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 $\text{eval}$ may return;

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

(c) the use(s) of environment(s) in $\text{letsta}$ and in a call to a function defined by $\text{letsta}$;

(d) the use(s) of environment(s) in $\text{letdyn}$ and in a call to a function defined by $\text{letdyn}$.

[8 marks]

Hint: because both $\text{letsta}$ and $\text{letdyn}$ 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.