1) Which of the following produce different results in a eager language and a lazy language? Both produce
the same result if they both produce the same number or they both produce a procedure (even if the
procedure doesn’t behave exactly the same when applied), but they can differ in errors reported.

   a) \{\lambda \{y\} 12\} \{1 2\}
   b) \lambda \{x\} \{\lambda \{y\} 12\} \{1 2\}\}
   c) \{+ 1 \lambda \{y\} 12\}
   d) \{+ 1 \{\lambda \{x\} \{+ 1 13\}\} \{+ 1 \lambda \{z\} 12\}\}
   e) \{+ 1 \{\lambda \{x\} \{+ x 13\}\} \{+ 1 \lambda \{z\} 12\}\}
2) Given the type rules

\[ \ldots x \leftarrow \tau \ldots \vdash x : \tau \quad \Gamma \vdash 1 : \text{num} \]

\[ \Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num} \]

\[ \frac{}{\Gamma \vdash \{+ e_1 e_2\} : \text{num}} \]

\[ \Gamma \vdash \{\lambda \{[x : \tau_1]\} e\} : (\tau_1 \rightarrow \tau_2) \quad \Gamma \vdash e_1 : (\tau_1 \rightarrow \tau_2) \quad \Gamma \vdash e_2 : \tau_1 \]

\[ \frac{}{\Gamma \vdash \{e_1 e_2\} : \tau_2} \]

in one of the following expressions, the \_\_\_\_ can be filled in with a type so that the resulting expression has a type in the empty environment, while there is no type for the \_\_\_\_ that causes the other to have a type. Pick the right expression and show a derivation tree (which is a trace of typecheck that's written in the style as the type rules above) demonstrating that the chosen expression has a type.

\{\{\lambda \{[x : \_\_\_\_]\} \{+ x 1\}\} x\}

\{\lambda \{[x : \_\_\_\_]\} \{+ \{x 1\} 1\}\}

Note that your answer should not include symbols like \(\Gamma\), \(\tau\), or \(e\), except when used as designated abbreviations, since those are meta-variables that are replaced by concrete environments, types, and expressions in the derivation tree.
3) Given the following expression:

\[
\{\lambda \{x\} \{x \ x\}\} \\
\{\lambda \{y\} \ 12\}\}
\]

Describe a trace of the evaluation in terms of arguments to \texttt{interp} and \texttt{continue} functions for every call of each in the \texttt{lambda-k.rkt} interpreter. (There will be 7 calls to \texttt{interp} and 5 calls to \texttt{continue}.) The \texttt{interp} function takes three arguments — an expression, an environment, and a continuation — so show all three for each \texttt{interp} call. The \texttt{continue} function takes two arguments — a continuation and a value — so show both for each \texttt{continue} call. Represent continuations using records.
Answers

1) $a$ and $d$.

2)

\[
\begin{align*}
\Gamma_1 \vdash x : (\text{num} \rightarrow \text{num}) & \quad \Gamma_1 \vdash 1 : \text{num} \\
\Gamma_1 \vdash \{x 1\} : \text{num} & \\
\Gamma_1 = [x \leftarrow (\text{num} \rightarrow \text{num})] \vdash \{\{x 1\} 1\} : \text{num} \\
\emptyset \vdash \{\lambda x : (\text{num} \rightarrow \text{num})\} \{\{x 1\} 1\} : (\text{num} \rightarrow \text{num}) \rightarrow \text{num}
\end{align*}
\]

3)

\[
\begin{align*}
\text{interp} \ \text{expr} &= \{\{\lambda x \{x x\}\} \{\lambda y \ 12\}\} \\
\text{env} &= \text{mt-env} \\
k &= (\text{doneK})
\end{align*}
\]

\[
\begin{align*}
\text{interp} \ \text{expr} &= \{\lambda x \{x x\}\} \\
\text{env} &= \text{mt-env} \\
k &= (\text{appArgK} \{\lambda y \ 12\} \text{mt-env} (\text{doneK}))
\end{align*}
\]

\[
\begin{align*}
\text{cont} \ k &= (\text{appArgK} \{\lambda y \ 12\} \text{mt-env} (\text{doneK})) \\
\text{val} &= (\text{closV} \ 'x \ {x x} \text{mt-env}) = v_1
\end{align*}
\]

\[
\begin{align*}
\text{interp} \ \text{expr} &= \{\lambda y \ 12\} \\
\text{env} &= \text{mt-env} \\
k &= (\text{doAppK} v_1 (\text{doneK}))
\end{align*}
\]

\[
\begin{align*}
\text{cont} \ k &= (\text{doAppK} v_1 (\text{doneK})) \\
\text{val} &= (\text{closV} \ 'y \ 12 \text{mt-env}) = v_2
\end{align*}
\]

\[
\begin{align*}
\text{interp} \ \text{expr} &= \{x \ x\} \\
\text{env} &= (\text{extend-env} (\text{bind} \ 'x \ v_2) \text{mt-env}) = e_1 \\
k &= (\text{doneK})
\end{align*}
\]

\[
\begin{align*}
\text{interp} \ \text{expr} &= x \\
\text{env} &= e_1 \\
k &= (\text{appArgK} x e_1 (\text{doneK}))
\end{align*}
\]

\[
\begin{align*}
\text{cont} \ k &= (\text{appArgK} x e_1 (\text{doneK})) \\
\text{val} &= v_2
\end{align*}
\]

\[
\begin{align*}
\text{interp} \ \text{expr} &= x \\
\text{env} &= e_1 \\
k &= (\text{doAppK} v_2 (\text{doneK}))
\end{align*}
\]

\[
\begin{align*}
\text{cont} \ k &= (\text{doAppK} v_2 (\text{doneK})) \\
\text{val} &= v_2
\end{align*}
\]

\[
\text{interp} \ \text{expr} = 12
\]
env = (extend-env (bind 'y v2) mt-env)
k = (doneK)
cont k = (doneK)
val = (numV 12)