Sample Mid-Term Exam 2 (take-home)

CS 6510, Spring 2017

actual exam due April 10

Name: ____________________

Start time: ____________________
End time: ____________________

Instructions: You have eighty minutes to complete this open-book, open-note, closed-computer exam. Please write all answers in the provided space, plus the back of the exam if necessary.

1) Which of the following produce different results in an 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

$$[\ldots \mathbf{x} \leftarrow \tau \ldots] \vdash \mathbf{x} : \tau \qquad \Gamma \vdash 1 : \texttt{num} \qquad \frac{\Gamma \vdash \mathbf{e}_1 : \texttt{num} \quad \Gamma \vdash \mathbf{e}_2 : \texttt{num}}{\Gamma \vdash \{\texttt{+}\ \mathbf{e}_1\ \mathbf{e}_2\} : \texttt{num}}$$

$$\frac{\Gamma[\mathbf{x} \leftarrow \tau_1] \vdash \mathbf{e} : \tau_2}{\Gamma \vdash \{\texttt{lambda}\ \{[\mathbf{x}\ \texttt{:}\ \tau_1]\}\ \mathbf{e}\} : (\tau_1 \rightarrow \tau_2)} \qquad \frac{\Gamma \vdash \mathbf{e}_1 : (\tau_1 \rightarrow \tau_2) \quad \Gamma \vdash \mathbf{e}_2 : \tau_1}{\Gamma \vdash \{\mathbf{e}_1\ \mathbf{e}_2\} : \tau_2}$$

in one of the following expressions, the `____` can be filled in with a type so that the resulting expression has a type in the enmpty 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 Γ, τ, 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 evalaution in terms of arguments to `interp` and `continue` functions for every call of each. (There will be 7 calls to `interp` and 5 calls to `continue`.) The `interp` function takes three arguments — an expression, an environment, and a continuation — so show all three for each `interp` call. The `continue` function takes two arguments — a value and a continuation — so show both for each `continue` call. Represent continuations using records.

4) Suppose a garbage-collected interepreter uses the following three kinds of records:

- Tag **1**: a record containing two pointers
- Tag **2**: a record containing one pointer and one integer
- Tag **3**: a record containing one integer

The interpreter has one register, which always contains a pointer, and a memory pool of size 22. The allocator/collector is a two-space copying collector, so each space is of size 11. Records are allocated consecutively in to-space, starting from the first memory location, 0.

The following is a snapshot of memory just before a collection where all memory has been allocated:

- Register: 8
- To space: 1 3 8 3 0 2 3 7 2 0 8

What are the values in the register and the new to-space (which is also addressed starting from 0) after collection? Assume that unallocated memory in to-space contains 0.

- Register:

- To space:

Answers

1) *a* and *d*.

2)

$$\dfrac{\dfrac{\dfrac{\Gamma_1 \vdash \texttt{x} : (\texttt{num} \rightarrow \texttt{num}) \quad \Gamma_1 \vdash \texttt{1} : \texttt{num}}{\Gamma_1 \vdash \{\texttt{x 1}\} : \texttt{num}} \qquad \Gamma_1 \vdash \texttt{1} : \texttt{num}}{\Gamma_1 = [\texttt{x} \leftarrow (\texttt{num} \rightarrow \texttt{num})] \vdash \{\texttt{+ } \{\texttt{x 1}\} \texttt{ 1}\} : \texttt{num}}}{\emptyset \vdash \{\texttt{lambda } \{[\texttt{x : } (\texttt{num} \rightarrow \texttt{num})]\}\} \ \{\texttt{+ } \{\texttt{x 1}\} \texttt{ 1}\}\} : ((\texttt{num} \rightarrow \texttt{num}) \rightarrow \texttt{num})}$$

3)

$$\begin{array}{lll} \text{interp} & \text{expr} & = \boxed{\{\{\texttt{lambda } \{\texttt{x}\} \ \{\texttt{x x}\}\} \ \{\texttt{lambda } \{\texttt{y}\} \texttt{ 12}\}\}} \\ & \text{env} & = \texttt{mt-env} \\ & \text{k} & = \texttt{(doneK)} \end{array}$$

$$\begin{array}{lll} \text{interp} & \text{expr} & = \boxed{\{\texttt{lambda } \{\texttt{x}\} \ \{\texttt{x x}\}\}} \\ & \text{env} & = \texttt{mt-env} \\ & \text{k} & = \texttt{(appArgK } \boxed{\{\texttt{lambda } \{\texttt{y}\} \texttt{ 12}\}} \texttt{ mt-env (doneK))} \end{array}$$

$$\begin{array}{lll} \text{cont} & \text{val} & = \texttt{(closureV 'x } \boxed{\{\texttt{x x}\}} \texttt{ mt-env)} = V_1 \\ & \text{k} & = \texttt{(appArgK } \boxed{\{\texttt{lambda } \{\texttt{y}\} \texttt{ 12}\}} \texttt{ mt-env (doneK))} \end{array}$$

$$\begin{array}{lll} \text{interp} & \text{expr} & = \boxed{\{\texttt{lambda } \{\texttt{y}\} \texttt{ 12}\}} \\ & \text{env} & = \texttt{mt-env} \\ & \text{k} & = \texttt{(doAppK } V_1 \texttt{ (doneK))} \end{array}$$

$$\begin{array}{lll} \text{cont} & \text{val} & = \texttt{(closureV 'y } \boxed{\texttt{12}} \texttt{ mt-env)} = V_2 \\ & \text{k} & = \texttt{(doAppK } V_1 \texttt{ (doneK))} \end{array}$$

$$\begin{array}{lll} \text{interp} & \text{expr} & = \boxed{\{\texttt{x x}\}} \\ & \text{env} & = \texttt{(extend-env (bind 'x } V_2\texttt{) mt-env)} = E_1 \\ & \text{k} & = \texttt{(doneK)} \end{array}$$

$$\begin{array}{lll} \text{interp} & \text{expr} & = \boxed{\texttt{x}} \\ & \text{env} & = E_1 \\ & \text{k} & = \texttt{(appArgk } \boxed{\texttt{x}} \ E_1 \texttt{ (doneK))} \end{array}$$

$$\begin{array}{lll} \text{cont} & \text{val} & = V_2 \\ & \text{k} & = \texttt{(appArgK } \boxed{\texttt{x}} \ E_1 \texttt{ (doneK))} \end{array}$$

$$\begin{array}{lll} \text{interp} & \text{expr} & = \boxed{\texttt{x}} \\ & \text{env} & = E_1 \\ & \text{k} & = \texttt{(doAppK } V_2 \texttt{ (doneK))} \end{array}$$

$$\begin{array}{lll} \text{cont} & \text{val} & = V_2 \\ & \text{k} & = \texttt{(doAppK } V_2 \texttt{ (doneK))} \end{array}$$

$$\begin{array}{lll} \text{interp} & \text{expr} & = \boxed{\texttt{12}} \end{array}$$

	env	=	`(extend-env (bind 'y` V_2`) mt-env)`
	k	=	`(doneK)`
cont	val	=	`(numV 12)`
	k	=	`(doneK)`

4) Register: 0, To space: 2 3 8 1 6 0 3 0 0 0 0