Class test 2

Write your name and secondary ID (of the form abc123) on the top left corner of all your answer sheets.

Question 1

Consider the following C code:

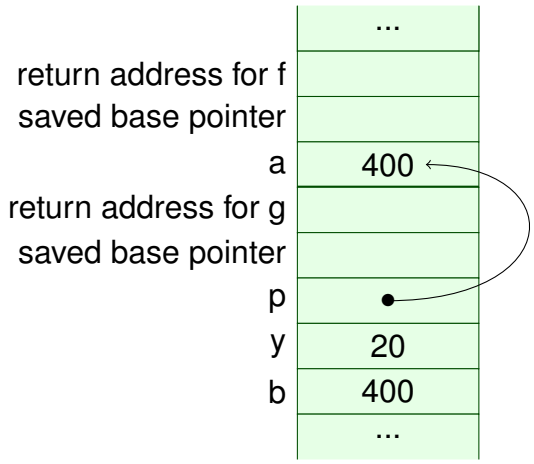
```
void g(long *p, long y)
{
  long b;
  b = y * y;
  *p = b;
}

long f()
{
  long a = 20;
  g(&a, a)
  return a;
}
```

Draw the call stack after `f` has called `g` and just before `g` returns, giving as much detail as you can. Then explain which variables *could* be kept in registers and which *must* be in the stack frame.

[40%]

Solution



The variable `b` could be kept in a register, but this is not possible for `a`, as it needs to be acessed using a pointer, as can be seen with the arrow in the diagram.

Question 2

Let E be some environment. Show the steps that the CEK machine performs starting from the following state: [40%]

$$\langle (\lambda z.z)\, 8 \;\mid E \mid \blacksquare \rangle$$

Solution

The CEK machine reduces the expression to 8 and then terminates.

$$\begin{aligned}
& \langle (\lambda z.z)\, 8 \;\mid E \mid \blacksquare \rangle \\
\rightsquigarrow\; & \langle \lambda z.z \;\mid E \mid (\bigcirc\ 8\ E)\,, \blacksquare \rangle \\
\rightsquigarrow\; & \langle \mathtt{clos}(\lambda z.z, E) \;\mid E \mid (\bigcirc\ 8\ E)\,, \blacksquare \rangle \\
\rightsquigarrow\; & \langle 8 \;\mid E \mid (\mathtt{clos}(\lambda z.z, E)\ \bigcirc)\,, \blacksquare \rangle \\
\rightsquigarrow\; & \langle z \;\mid E[z \mapsto 8] \mid \blacksquare \rangle \\
\rightsquigarrow\; & \langle 8 \;\mid E[z \mapsto 8] \mid \blacksquare \rangle
\end{aligned}$$

Question 3

Consider the following expression

$$(\lambda x.\lambda f.(f\ 0))\ 1\ ((\lambda x.\lambda y.x)\ 2)$$

Note that there are two different bound variables x. The first x is passed the actual parameter 1 and the second x is passed 2. Explain what this expression evaluates to in the CEK machine. You do not need to compute the complete run, but you should explain how closures are used to get the correct value of x.

[20%]

Solution

The important things to note are the following. At the point of call, $f\ 0$, the environment binds x to the value 1. However, $(\lambda y.x)$ evaluates to a closure in which the environment binds x to 2. The latter environment is used when the function call to f is evalued. Omitting some steps, here is how the closures are built:

$$
\begin{array}{rl}
 & \langle (\lambda x.\lambda f.f\ 0)\ 1\ ((\lambda x.(\lambda y.x))\ 2)\ \mid \emptyset \mid \blacksquare \rangle \\
\rightsquigarrow \cdots \rightsquigarrow & \langle \texttt{clos}(\lambda f.f\ 0, x \mapsto 1)\ \mid \emptyset \mid (\bigcirc\ ((\lambda x.(\lambda y.x))\ 2)\ \emptyset)\ , \blacksquare \rangle \\
\rightsquigarrow \cdots \rightsquigarrow & \langle \texttt{clos}(\lambda y.x, x \mapsto 2)\ \mid x \mapsto 2 \mid (\texttt{clos}(\lambda f.f\ 0, x \mapsto 1)\ \bigcirc), \blacksquare \rangle \\
\rightsquigarrow \cdots \rightsquigarrow & \langle x\ \mid x \mapsto 2, y \mapsto 0 \mid \blacksquare \rangle \\
\rightsquigarrow & \langle 2\ \mid x \mapsto 2, y \mapsto 0 \mid \blacksquare \rangle
\end{array}
$$