1 Reduction and variable capture

1. Answer:

   Non capture avoiding

   $$\begin{aligned}&(\lambda x.(\lambda y.x))\ y\\ &=_\beta \lambda y.y\end{aligned}$$

   Capture avoiding

   $$\begin{aligned}&(\lambda x.(\lambda y.x))\ y\\ &=_\alpha (\lambda x.(\lambda z.x))\ y\\ &=_\beta \lambda z.y\end{aligned}$$

2. Answer:

   Non capture avoiding

   $$\begin{aligned}&(\lambda x.(\lambda y.x))\ (\lambda y.x)\\ &=_\beta \lambda y.(\lambda y.x)\end{aligned}$$

   Capture avoiding

   $$\begin{aligned}&(\lambda x.(\lambda y.x))\ (\lambda y.x)\\ &=_\alpha (\lambda x.(\lambda z.x))\ (\lambda y.x)\\ &=_\beta \lambda zy.x\end{aligned}$$

3. Answer:

   Non capture avoiding

   $$\begin{aligned}&(\lambda x.(\lambda y.x))\ (\lambda y.y)\\ &=_\beta \lambda y.(\lambda y.y)\end{aligned}$$

   Capture avoiding

   $$\begin{aligned}&(\lambda x.(\lambda y.x))\ (\lambda y.y)\\ &=_\alpha (\lambda x.(\lambda z.x))\ (\lambda y.y)\\ &=_\beta \lambda zy.y\end{aligned}$$

4. Answer:

Non capture avoiding

$$\begin{aligned}
&(\lambda xyz.\ \lambda fgh.\ f\ x\ (g\ y)\ (h\ z))\ h\ (\lambda ab.\ a\ (g\ b))\ f\\
&=_\beta (\lambda yz.\ \lambda fgh.\ f\ h\ (g\ y)\ (h\ z))\ (\lambda ab.\ a\ (g\ b))\ f\\
&=_\beta (\lambda z.\ \lambda fgh.\ f\ h\ (g\ (\lambda ab.\ a\ (g\ b)))\ (h\ z))\ f\\
&=_\beta \lambda fgh.\ f\ h\ (g\ (\lambda ab.\ a\ (g\ b)))\ (h\ f)
\end{aligned}$$

Capture avoiding

$$\begin{aligned}
&(\lambda xyz.\ \lambda fgh.\ f\ x\ (g\ y)\ (h\ z))\ h\ (\lambda ab.\ a\ (g\ b))\ f\\
&=_\alpha (\lambda xyz.\ \lambda qgr.\ (q\ x)\ (g\ y)\ (r\ z))\ h\ (\lambda ab.\ a\ (g\ b))\ f\\
&=_\beta (\lambda yz.\ \lambda qgr.\ (q\ h)\ (g\ y)\ (r\ z))\ (\lambda ab.\ a\ (g\ b))\ f\\
&=_\beta (\lambda z.\ \lambda qgr.\ (q\ h)\ (g\ (\lambda ab.\ a\ (g\ b)))\ (r\ z))\ f\\
&=_\beta \lambda qgr.\ (q\ h)\ (g\ (\lambda ab.\ a\ (g\ b)))\ (r\ f)
\end{aligned}$$

5. Answer:

$$\begin{aligned}
&(\lambda xy.z)\ z\ z\\
&= (\lambda xy.z)\ (z\ z)\\
&= \lambda y.z
\end{aligned}$$

6. Answer:

$$\begin{aligned}
&(\lambda x.(\lambda y.(x\ y)))\\
&=_\eta \lambda x.x
\end{aligned}$$

7. Answer:

$$\begin{aligned}
S &= \lambda xyz.((x\ z)\ (y\ z))\\
K &= \lambda xy.x\\
I &= \lambda x.x
\end{aligned}$$

S K S = (S K) S by expression left association. The following are done with call-by-name.

$$\begin{aligned}
S\ K &= (\lambda xyz.((x\ z)\ (y\ z)))(\lambda xy.x)\\
&=_\beta (\lambda yz.(((\lambda xy.x)\ z)\ (y\ z)))\\
&=_\beta (\lambda yz.((\lambda y.z)\ (y\ z)))\\
&=_\beta \lambda yz.z
\end{aligned}$$

$$(S\ K)\ S = (\lambda yz.z)\ (\lambda xyz.((x\ z)\ (y\ z))) \quad = \beta(\lambda z.z) = \alpha(\lambda x.x) = I$$

S K S hence shown reduces to I.

2 Variable Binding and Closure

1. Answer: The function is recursive and never terminates. This is because the x(2) call on line 2 refers to the definition of x on line 2 which makes it recursive.

2. Answer: x passed in as argument on line two binds to the previous definition on line 1 because at the time of reference, x on line 2 is not assigned yet since the function is not executed at that point.

3. Answer: You get the following error: Error reassigning x. This is because Haskel use static types. Variables are not allowed to change values.

3 Closure and access links

1. See scaned document

2. 18

3. This does not terminate because the function is infinitely recursive without a base case.

4 Memory management and high-order functions

1. See scaned document

2. The value is 10 because. Within function scope g, x is set to 7. This is passed in as argument to h which is f. Therefore the y argument in f resolves to 7. x is 5 by access link to the parent scope. The function f therefore returns 5 + 7 - 2 which is 10.

5 More subsitution and variable capture

1. Answer:

$$
\begin{aligned}
&((\lambda x.((\lambda x.x)\ 2) + x)\ x)[x := 3] \\
&=_\beta (((\lambda x.x)\ 2) + x)[x := 3] \\
&=_\beta (2 + x)[x := 3] \\
&= 2 + 3 \\
&= 5
\end{aligned}
$$

Explaination: In this case it would not matter because the only x that is not captured is one that is ok to replace. All other x are captured by lambda expressions. 3 can not possibly be captured by any lambda.

2. Answer:

$$
\begin{aligned}
&(\lambda y.(\lambda xyz.z)\ y\ z\ y)[z := w] \\
&= (\lambda y.((((\lambda xyz.z)\ y)\ z)\ y))[z := w] \\
&=_\beta (\lambda y.(((\lambda yz.z)\ y)\ z))[z := w] \\
&=_\beta (\lambda y.((\lambda z.z)\ y))[z := w] \\
&=_\beta (\lambda y.y)[z := w] \\
&= \lambda y.y
\end{aligned}
$$

Explaination: In this case it would not matter whether capture avoiding is taken into account. w is not caputured anywhere, therefore subsitution is legal in this case without needing to avoid capturing w.

3. Answer:

$$
\begin{aligned}
&(\lambda p.(\lambda x.p(x\ x))(\lambda x.p))[x := p] \\
&=_\alpha (\lambda z.(\lambda x.(z(x\ x)))(\lambda x.z))[x := p] \\
&=_\beta (\lambda x.((\lambda x.z)(x\ x)))[x := p] \\
&=_\beta (\lambda x.z)[x := p] \\
&=_\beta (\lambda x.z)
\end{aligned}
$$

Explaination: Yes, capture avoiding is a must because p is captured by the lambda expression. This violates subsitution rules therefore capture avoiding (alpha renaming) must be performed.