

1. What are the three kinds of λ -calculus terms
2. What is the closest λ -calculus analogue to function creation.
3. What is the closest λ -calculus analogue to calling a function.
4. What is the difference between a free and bound variable?
5. List the free variables in the following terms:
 - (a) x
 - (b) $\lambda x.x$
 - (c) $\lambda x.y$
 - (d) $\lambda y.(xz)y$
 - (e) $\lambda x.\lambda y.(xz)y$
 - (f) $\lambda x.(\lambda y.xz)y$
6. For each pair of Lambda terms, state whether they are equivalent under α -renaming:
 - (a) x y
 - (b) $\lambda x.x$ $\lambda y.y$
 - (c) $\lambda x.x$ $\lambda y.x$
 - (d) $\lambda x.x$ $\lambda x.y$
 - (e) $\lambda x.y$ $\lambda y.y$
 - (f) $\lambda x.\lambda y.xy$ $\lambda a.\lambda b.ab$
 - (g) $\lambda x.\lambda y.xy$ $\lambda x.\lambda x.xx$
7. What is the result of applying one-step of β -reduction to the following:
 - (a) $(\lambda x.x)(\lambda y.\lambda z.z)$
 - (b) $\lambda x.(\lambda y.x)(\lambda z.zz)$
 - (c) $(\lambda x.ax)((\lambda y.b)(\lambda z.cz))$
 - (d) $(\lambda x.xx)(\lambda y.yy)$
8. Currying
 - (a) What is currying?
 - (b) Write a curried λ calculus function that when applied to two values (in sequence), evaluates to the first.
 - (c) Write a curried λ calculus function that when applied to two values (in sequence), evaluates to the second.
9. What is the normal form of a λ -expression?
10. What does the Church-Rosser Theorem (confluence) tell us about the relationship between equivalent λ -terms and normal forms?

11. Consider: $((\lambda f.\lambda g.f(ga))(\lambda x.bx))(\lambda y.cy)$ (c.f., Exercise 4.4 in Mitchell, p. 83)
- Reduce the expression by choosing, at each step, the reduction that eliminates a λ as far to the *left* as possible. (Show each reduction step.)
 - Reduce the expression by choosing, at each step, the reduction that eliminates a λ as far to the *right* as possible. (Show each reduction step.)
12. For each of the following, if there is normal form, give the normal form, if there is no normal form, show one β -reduction step:
- $(\lambda x.\lambda y.xy)(\lambda z.z)$
 - $(\lambda x.xx)(\lambda x.xx)$
 - $((\lambda x.\lambda y.y)(\lambda a.a))(\lambda b.b)$
 - $(\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))(\lambda x.xx)$
 - $((\lambda x.\lambda y.xy)(\lambda a.a))(\lambda b.b)$
13. Church numerals:
- What are the Church numerals?
 - How would you represent the computation of $2 + 3$ in Church numerals?
 - Show the process of reducing the λ -term for $2 + 2$ to the term for 4.
14. The Church booleans are a way of representing booleans by using the function $\lambda x.\lambda y.x$ as TRUE and the function $\lambda x.\lambda y.y$ as FALSE
- Consider the function $\lambda b.\lambda x.\lambda y.byx$ and what it evaluates to when applied to TRUE or FALSE (e.g. What does $(\lambda b.\lambda x.\lambda y.byx)(\lambda x.\lambda y.x)$ reduce to?)
- What do you think the function $\lambda b.\lambda x.\lambda y.byx$ does?
- (If you use some reference material to find this; please cite your source.)
15. What is Y? Write the λ -term and explain how it relates to recursion.
16. What is a λ -calculus expression that reduces to itself?
17. Explain the behavior of the program: (c.f., Exercise 4.6 in Mitchell, p. 84)
- ```
int f(int (*g)(...)){
 return g(g);
}
int main() {
 int x;
 x = f(f);
 printf("Value of x = %d\n", x);
 return 0;
}
```
18. **Extra credit:** Write a Scheme expression that evaluates to the list representing itself. Note, you can use the scheme function `eval` to test this; if your expression is *expr* then *expr*, (`eval expr`), (`eval (eval expr)`), (`eval (eval (eval expr))`) would all evaluate to the same thing. Hint: you'll probably need to make use of `lambda`, `cons` (or `list`), and `quote`.