CS131 Lambda Calculus Worksheet
This isn’t homework and is worth no credit. But it’s good practice!

Name: _________________________________________________________________

CAS ID (e.g., abc01234@pomona.edu): ______________________________________

I encourage you to collaborate. Please record your collaborations below.

Except for the problems in Section 7, you should use equational reasoning rather than CBN or CBV reduction.

Most solutions can be written in a single-line. Some solutions may take as many as four or five lines, but any more and you’re off the scent.

Unless the problem says otherwise, feel free to use any definition from class or in the lecture notes, like true and succ.

Problems marked with a (C) are “challenge” problems—go ahead and test your mettle, but these are longer or harder than anything I’d put on an exam.

Collaborators: ___________________________________________________________
1 Substitution

1.1 Changing names

Substitute $y$ for $x$ in $\lambda z. x$, i.e., compute $(\lambda z. x)[y/x]$.

1.2 Accept no substitutes

Substitute $y$ for $x$ in $\lambda z. z$, i.e., compute $(\lambda z. z)[y/x]$.

1.3 Identity crisis

Substitute $\lambda y. y$ for $x$ in $\lambda x. x$, i.e., compute $(\lambda x. x)[\lambda y. y/x]$.

1.4 Don’t get carried away

Substitute $\lambda x. x x$ for $x$ in $\lambda x. x x$, i.e., compute $(\lambda x. x x)[\lambda x. x x/x]$.
2 Alpha renaming and beta reduction

2.1 Explicit content
Use $=_{\beta}$ to reduce $\lambda x. y. x \ (\lambda y. y) \ (\lambda x. x)$ as much as possible. Use $=_{\alpha}$ to avoid potential name conflicts.

Perform the same reduction without using $=_{\alpha}$.

2.2 I come from a land down under
Reduce $\lambda x. (\lambda y. y) \ x$ as much as possible.

2.3 You can count on me
Reduce $\lambda x. z. s (s (s z)) \ (\lambda x. i. x) \ n$ as much as possible.
3  Booleans

3.1  Ask not what not can do for you, but what you can do for not

Write not (a/k/a negation, ¬, !) on Church booleans without using anything other than lambdas, variables, and expressions.

Write not a different way, using existing definitions.

3.2  Either/or

Write xor (a/k/a exclusive-or, ⊕, ^) on Church booleans without using anything other than lambdas, variables, and expressions.

Write xor a different way, using existing definitions.
3.3 Any boolean you like

Write a function corresponding to the predicate \( P(x, y, z) = (x \land y) \lor (\neg x \land z) \lor (\neg x \land \neg y \land \neg z) \).

3.4 Choices, choices

Write a function of two arguments that (a) returns its first argument if its second argument is true and (b) returns \( \lambda x. x \) otherwise.
4 Church numerals

4.1 Don’t be a □
Write a function that takes in a Church numeral \( n \) and squares it, returning \( n^2 \).

4.2 Polynomial want a cracker?
Write a function that takes in Church numerals \( x \), \( y \), and \( z \) and returns the value of \( y^3 + 3x^2z^2 + 2z + 5 \).

4.3 With great power comes great responsibility
Write exponentiation, i.e., a function that takes as arguments two Church numerals \( m \) and \( n \) and returns \( m^n \).
4.4 Toe the line

Write a function that takes in a slope $m$, a y-intercept $b$, and a Church pair of numbers $x$ and $y$; return true if $y = mx + b$ and false otherwise.

4.5 Binomial, save later (C)

Write a function `binomial` that takes Church numerals $x$, $y$, and $n$ and computes $(x + y)^n$ according to the Binomial Theorem:

$$(x + y)^n = \sum_{j=0}^{n} \binom{n}{j} x^{n-j} y^j$$

You can assume you have `choose` from Problem 5.2. Don’t use the Y combinator.
5  Recursion

5.1  Why ask Y?
Write the Fibonacci function.

5.2  I choose you, Pikachu!
Write a function choose that takes Church numerals \(m\) and \(n\) and computes \(\binom{m}{n}\). You’ll need division, but you can do this problem without having solved Problem 5.3—just assume you have divide.

It was super effective!
5.3 Divide and conquer (C)

Write divide on Church numerals.

Write a function divides that checks divisibility, i.e., it takes two Church numerals $m$ and $n$ and returns true if $m$ divides $n$ and false otherwise.
5.4 Go ahead and be negative (C)

Church numerals represent the natural numbers \( \mathbb{N} = \{0, 1, \ldots\} \). Use Church numerals, ingenuity, and elbow grease to represent the integers \( \mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\} \). Define \( \text{zzero} \), \( \text{zsucc} \), \( \text{ziszero} \), \( \text{zpred} \), \( \text{zplus} \), \( \text{zminus} \), \( \text{ztimes} \), and \( \text{zequal} \).
6 Lists

Let’s define lists as follows:

\[
\begin{align*}
\text{nil} &= \lambda n \ c. \ n \\
\text{cons} &= \lambda h \ t. \ \lambda n \ c. \ c \ h \ t \\
\text{null} &= \lambda l. \ \text{true} (\lambda h \ t. \ \text{false}) \\
\text{head} &= \lambda l. \ \Omega (\lambda h \ t. \ h) \\
\text{tail} &= \lambda l. \ \Omega (\lambda h \ t. \ t) \\
\text{foldr} &= Y (\lambda \text{foldr}. \ \lambda f \ b \ l. \ l \ b (\lambda h \ t. \ f \ h (\text{foldr} \ f \ b \ t)))
\end{align*}
\]

6.1 A one, a two, a one two three four

Write down the list [1, 2, 3, 4], i.e., the list containing the Church numerals one, two, three, and four in that order. Use nil and cons.

Write down the same list “directly”: don’t use anything but lambdas, variables, applications, and the Church numerals.

6.2 Two heads are better than one

Write down a function that returns the second element of a list. Your function should diverge if the list is too short.
6.3 One-by-one
Write \texttt{map} without using \texttt{foldr}.

Write \texttt{map} using \texttt{foldr}.

6.4 To the left, to the left
Write \texttt{foldl}.
6.5 I prefer French press
Write filter without using foldr.

Write filter using foldr.

6.6 Line ‘em up
Write a function that takes a number $n$ and produces the list $[1, 2, 3, \ldots, n]$. Return the empty list if $n$ is zero.
6.7 Panning for gold (C)

Write a function that takes a number \( n \) and returns a list of all primes less than or equal to \( n \). You’ll need the `divides` predicate from Problem 5.3 in addition to the list functions we’ve just defined. Good luck!
7 You say CBN, I say CBV—let’s call the whole thing off

7.1 Stop right there

How many steps does it take \((\lambda x \ y \ y) \ (\text{succ\ zero}) \ (\text{succ\ one})\) to reduce to a value using call-by-name? Show your work.

How many steps does it take \((\lambda x \ y \ y) \ (\text{succ\ zero}) \ (\text{succ\ one})\) to reduce to a value using call-by-value? Show your work.

7.2 One of these things is not like the other

Write a term that diverges in CBV but not in CBN.

7.3 Common ground

Write a term that diverges in both CBV and CBN. Don’t just write \(\Omega\): the term should take at least three steps in either evaluation scheme before looping.
7.4 CBV Church booleans (C)

In CBV evaluation, we evaluate all of a function’s arguments before $\beta$-reducing. Define a version of the Church booleans (true, false, and, or, and not) and some syntactic sugar for if expressions that work in CBV evaluation. For example, the syntactic sugar for let expressions is let $x = e_1$ in $e_2 \equiv (\lambda x. e_2) e_1$. 
7.5 Loop the loop (C)

In CBV, \( \Omega = (\lambda x. x x) (\lambda x. x x) \) reduces to itself in one step. Write a term that reduces to itself in two steps.

Write a term that reduces to itself in three steps.

Given a number \( n > 1 \), what would a term that reduces to itself in \( n \) steps look like?

Does your term also work for CBN? If not, write one that does.