1. Let $M_1$ be a closed and well-typed PCF term that is neither a lambda abstraction nor a pair. Show that if $M_1 \xrightarrow{\text{left}} N_1$ then $M_1 \xrightarrow{\text{lazy}} N_1$. (Note: Your proof must be as detailed as the one shown in class. In particular, it should be clear if a step follows from an assumption or from the inductive hypothesis).

Proof: By induction on derivations of the form $M_1 \xrightarrow{\text{left}} N_1$

(i) if $M_1 = M + N$

(a) Assume $M + N \xrightarrow{\text{left}} M_1 + N$ because $M \xrightarrow{\text{left}} M_1$. By Ind, $M \xrightarrow{\text{lazy}} M_1$. By add left, then $M + N \xrightarrow{\text{lazy}} M_1 + N$

(b) Assume $M + N \xrightarrow{\text{left}} M_1 + N$ because $N \xrightarrow{\text{left}} N_1$ and $M$ is a normal form. By Ind, $N \xrightarrow{\text{lazy}} N_1$. By assumption that $M_1$ is closed, so $M$ is closed and also $M$ is a normal form. So $M$ can be only a numeral, then $M + N \xrightarrow{\text{lazy}} M + N_1$

(c) Assume $M \equiv m, N \equiv n$, then this holds since the same axiom is in both reduction strategies.

(ii) if $M_1 = \text{Eq?}MN$

(a) Assume $\text{Eq?}MN \xrightarrow{\text{left}} \text{Eq?}M_1N$ because $M \xrightarrow{\text{left}} M_1$. By Ind, $M \xrightarrow{\text{lazy}} M_1$. By subterm rules of lazy, then $\text{Eq?}MN \xrightarrow{\text{lazy}} \text{Eq?}M_1N$

(b) Assume $\text{Eq?}MN \xrightarrow{\text{left}} \text{Eq?}M_1N$ because $N \xrightarrow{\text{left}} N_1$ and $M$ is a normal form. By Ind, $N \xrightarrow{\text{lazy}} N_1$. By assumption that $M_1$ is closed, so $M$ is closed and also $M$ is a normal form. So $M$ can be only a numeral, then $\text{Eq?}MN \xrightarrow{\text{left}} \text{Eq?}MN_1$

(c) Assume $M \equiv m, N \equiv n$, then this holds since the same axiom is in both reduction strategies.

(iii) if $M_1 = \text{if } M \text{ then } N \text{ else } P$

(a) Assume $\text{if } M \text{ then } N \text{ else } P \xrightarrow{\text{left}} \text{if } M_1 \text{ then } N \text{ else } P$ because $M \xrightarrow{\text{left}} M_1$. By Ind, $M \xrightarrow{\text{lazy}} M_1$. By the rule under bool, then $\text{if } M \text{ then } N \text{ else } P \xrightarrow{\text{lazy}} \text{if } M_1 \text{ then } N \text{ else } P$

(b) Assume $\text{if } M \text{ then } N \text{ else } P \xrightarrow{\text{left}} \text{if } M \text{ then } N_1 \text{ else } P$ because $N \xrightarrow{\text{left}} N_1$ and $M$ is a normal form. By assumption that $M_1$ is closed, so $M$ is closed and so $M$ is either true or false. It should apply axiom rule. So it will conflict with the assumption. So it is impossible.

(c) Assume $\text{if } M \text{ then } N \text{ else } P \xrightarrow{\text{left}} \text{if } M \text{ then } N \text{ else } P_1$ because $P \xrightarrow{\text{left}} P_1$ and $M, N$ is a normal form. By assumption that $M_1$ is closed, so $M$ is closed and so $M$ is
either true or false. It should apply axiom rule. So it will conflict with the assumption. So it is impossible.

(iv) if \( M_1 = \text{Proj}_j M \) Assume \( \text{Proj}_j M \xrightarrow{\text{left}} \text{Proj}_j M_1 \) because \( M \xrightarrow{\text{left}} M_1 \). By Ind, \( M \xrightarrow{\text{lazy}} M_1 \). By the rule under pair, then \( \text{Proj}_i M \xrightarrow{\text{left}} \text{Proj}_i M_1 \)

(v) if \( M_1 = MN \)

(a) Assume \( MN \xrightarrow{\text{left}} M_1 N \) because \( M \xrightarrow{\text{left}} M_1 \). By Ind, \( M \xrightarrow{\text{lazy}} M_1 \). By the rule under functions, then \( MN \xrightarrow{\text{left}} M_1 N \)

(b) Assume \( MN \xrightarrow{\text{left}} MN_1 \) because \( N \xrightarrow{\text{left}} N_1 \) and \( M \) is a normal form. By assumption, \( M_1 \) is closed, so \( M \) is closed, and \( M \) can only be a lambda abstraction. So it should apply axiom rule (\( \beta \) reduction). Thus, it conflicts with the assumption. So it is impossible.

(vi) In base case, assume \( M_1 \xrightarrow{\text{left}} N_1 \) because \( M_1 \rightarrow N_1 \) using axiom rules. By ind, \( M_1 \xrightarrow{\text{lazy}} N_1 \).

2. Exercise 2.4.17 on page 91. Desugar the program:

(a) \( (\lambda f : \text{nat} \rightarrow \text{nat}.(\lambda g : \text{nat} \rightarrow \text{nat}.f(g5))(fxF))(\lambda x : \text{nat}.3) (\beta \text{red}) \)

(b) \( \\
\rightarrow (((\lambda g : \text{nat} \rightarrow \text{nat}.\lambda x : \text{nat}.3)g5)(fxF) (\beta \text{red}) \)

(c) \( \\
\rightarrow (\lambda x : \text{nat}.3)(fxF) (\beta \text{red}) \)

(d) \( \\
\rightarrow 3 \)

According to above \( \beta \) reduction and left-red axioms

\[
\begin{align*}
M &\rightarrow N \\
M \xrightarrow{\text{left}} N
\end{align*}
\]

we can also get \( a \xrightarrow{\text{left}} b \xrightarrow{\text{left}} c \xrightarrow{\text{left}} d \). So the deterministic evaluation halts with value 3 on this program.

3. Exercise 2.4.25 on page 96.

\[
YM \equiv (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))(\lambda x.\lambda y.y)
\]

\[
\xrightarrow{\text{left}} (\lambda x.(\lambda x.\lambda y.y)(xx))(\lambda x.(\lambda x.\lambda y.y)(xx))
\]

\[
\xrightarrow{\text{left}} (\lambda x.\lambda y.y)(\lambda x.(\lambda x.\lambda y.y)(xx))
\]

\[
\xrightarrow{\text{left}} \lambda y.y
\]

\[
ZM \equiv (\lambda f.(\lambda x.f(\text{delay}[xx])))(\lambda x.f(\text{delay}[xx])))(\lambda x.\lambda y.y)
\]

\[
\xrightarrow{\text{eager}} (\lambda x.(\lambda x.\lambda y.y)(\text{delay}[xx])))(\lambda x.(\lambda x.\lambda y.y)(\text{delay}[xx]))
\]

\[
\xrightarrow{\text{eager}} (\lambda x.\lambda y.y)(\text{delay}[\lambda x.(\lambda x.\lambda y.y)(\text{delay}[xx])])(\lambda x.(\lambda x.\lambda y.y)(\text{delay}[xx]))
\]

\[
\xrightarrow{\text{eager}} (\lambda x.\lambda y.y)(\lambda z.(\lambda x.(\lambda x.\lambda y.y)(\text{delay}[xx])))(\lambda x.(\lambda x.\lambda y.y)(\text{delay}[xx]))z)
\]

2
(Because eager reduction has no rule for lambda abstraction. a lambda abstraction is a value. Thus we can use \(\beta\) reduction directly)

\[
Y M \equiv (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))(\lambda x.\lambda y.y)
\]

When apply eager reduction to \(YM\), it will reduce infinitely and can not halt.

4. Exercise 2.6.1 on page 124.

For any sequence of type \(\sigma_1, \sigma_2, \ldots, \sigma_n\), we introduce the n-ary sum notation

\[
\sigma_1 + \cdots + \sigma_n \overset{\text{def}}{=} (\sigma_1 + (\cdots (\sigma_n - 1 + \sigma_n) \cdots))
\]

and introduce a function

\[
In_{\sigma_1 + \cdots + \sigma_n} : \sigma_i \rightarrow \sigma_1 + \cdots + \sigma_n
\]

And define \(In_{\sigma_1 + \cdots + \sigma_n} \overset{\text{def}}{=} \)

\[
\begin{cases}
\lambda x : \sigma_i : (In_{\sigma_1 + \cdots + \sigma_{i-1}} Inleft \ x) (i < n) \\
\lambda x : \sigma_n : In_{\sigma_1 + \cdots + \sigma_{n-1}} x (i = n)
\end{cases}
\]

And we can get: Case\(^{\sigma_1 + \cdots + \sigma_n} M_i\) \(f_1 \cdots f_n\) \(\overset{\text{def}}{=} \)

Case\(^{\sigma_1 + \cdots + \sigma_n} \overset{\text{def}}{=} \)

\[
\begin{align*}
&= f_1 \cdots f_n \\
&\quad (\lambda x : (\sigma_2 + \cdots + \sigma_n) \text{ case}^{\sigma_2, \sigma_3 + \cdots + \sigma_n} x f_2) \\
&\quad (\lambda x : (\sigma_3 + \cdots + \sigma_n) \text{ case}^{\sigma_3, \sigma_4 + \cdots + \sigma_n} x f_3) \\
&\quad \cdots \\
&\quad (\lambda x : (\sigma_n - 1 + \sigma_n) \text{ case}^{\sigma_n-1, \sigma_n} x f_{n-1} f_n)) \\
&\quad = f_1 M_i
\end{align*}
\]

Use above notation, we can translate n-ary sum type. We can choose some ordering of the label \(l_1, \cdots, l_k\) and write each type expression and record expression using this order. Then we can translate expression with variant into PCF as follow:

\[
[l_1 : \sigma_1 \cdots l_k : \sigma_k] \overset{\text{def}}{=} \sigma_1 + \cdots + \sigma_n
\]

\[
i[l_1 : \sigma_1 \cdots l_k : \sigma_k](M_i) \overset{\text{def}}{=} In_{\sigma_1 + \cdots + \sigma_k} M_i
\]
that is

\[
\begin{cases}
Inright^{i-1}Inleft \ x \ (i < k) \\
Inright^{k-1}x \ (i = k)
\end{cases}
\]

\[
\text{Case}^{l_1;\sigma_1;\cdots;l_k;\sigma_k}/\rho \sigma_i(M_i)f_1/\cdots/f_k \\
\overset{\text{def}}{=} \text{Case}(\sigma_1+\cdots+\sigma_k)\rho(In_1\sigma_1+\cdots+\sigma_k)M_if_1\cdots f_k
\]

that is

= \ f_iM_i