We can encode tuples and extraction of tuples with the Lambda Calculus in the following manner:

\[
\begin{align*}
Pair &= \lambda f . \lambda s . \lambda b . ((b f)s) \\
Fst &= \lambda p . (p \text{ True}) \\
Snd &= \lambda p . (p \text{ False})
\end{align*}
\]

where \text{True} and \text{False} are symbols predefined from class. The way to read the above is that \textit{Pair} takes two arguments (\textit{f} and \textit{s}) and makes an “object” (actually a function, as everything is) which is like an ML tuple containing \textit{f} and \textit{s}. \textit{Fst} and \textit{Snd} are functions that take a pair and return the first and second item, respectively. It’s equivalent to

\begin{verbatim}
fun Pair(f, s) = (f, s);
fun Fst(p) = #1(p);
fun Snd(p) = #2(p);
\end{verbatim}

Confirm that

\[Fst \ (Pair \ M \ N) \rightarrow_* \ M\]

that is, for expressions \textit{M} and \textit{N}, the expression on the left \(\beta\)-reduces (after one or more steps) to \textit{M}. You may assume, as we showed in class, that \(\text{True} \ X \ Y \rightarrow_* \ X\).

(Fully parenthesized, the expression on the left is \((Fst \ ((Pair \ M) \ N))\).)