[Ex. 1] Consider the HOFL term
\[ t \overset{\text{def}}{=} \text{rec } f. \lambda x. \text{if } x \text{ then } (x, \text{fst}(f \ x)) \text{ else } (1, 2) \]

1. Find the principal type of \( t \).
2. Compute the canonical form of the term \( \text{fst}(t \ 0) \).
3. Compute the (lazy) denotational semantics of \( t \).

[Ex. 2] Let \( \simeq \) denote strong bisimilarity and \( \approx \) be weak bisimilarity. Given an action \( \alpha \):

1. Find two CCS processes \( p, q \) such that
   \[ (p|q)\backslash\alpha \not\simeq p\backslash\alpha|(q\backslash\alpha) \quad \text{but} \quad (p|q)\backslash\alpha \approx p\backslash\alpha|(q\backslash\alpha) \]
2. Find two (non inactive) CCS processes \( r, s \) such that
   \[ (r|s)\backslash\alpha \simeq r\backslash\alpha|(s\backslash\alpha) \]

[Ex. 3] Two elevators \( e_1, e_2 \) serve three floors \( f_1, f_2, f_3 \). Consider the atomic propositions (with with \( i \in [1, 2] \) and \( j \in [1, 3] \)):
   - \( \text{at}_{i,j} \): holds when elevator \( e_i \) is at floor \( f_j \);
   - \( \text{up}_i \): holds when elevator \( e_i \) is moving upwards;
   - \( \text{down}_i \): holds when elevator \( e_i \) is moving downwards.

1. Write the property “whenever \( e_1 \) moves up, it will stop at \( f_2 \) or \( f_3 \) before it can move down” in LTL.
2. Write the property “it is possible to find both elevators at \( f_2 \)” in CTL.
3. Write the property “it is always the case that if \( e_1 \) moves up then \( e_2 \) goes down or is at \( f_1 \)” in the \( \mu \)-calculus.

[Ex. 4] A process is \textit{sequential} if it is written without using parallel composition. Consider the \( \pi \)-calculus process (with \( x, y, z \) pairwise different)
\[ p \overset{\text{def}}{=} \text{\textbar} y . \text{nil} \mid x(z) . \text{nil} \]

Write a sequential process \( q \) that is strongly (early and late) bisimilar to \( p \).

[Ex. 5] Write a GoogleGo function \texttt{Split} that takes three channels \( c, c_1, c_2 \) for passing integers and forwards each message arriving on \( c \) to either \( c_1 \) or \( c_2 \) depending on which one is ready to receive. When \( c \) is closed, \texttt{Split} closes \( c_1 \) and \( c_2 \) and terminates. When writing the function, type the channels according to their usages.