1. (8 pts) OCaml Types and Type Inference

   a. (2 pts) Give the type of the following OCaml expression

   \[
   \text{fun } x \ y \to (y + 2, \ x) \quad \text{Type } = \ 'a \to \text{int} \to \text{int} \times 'a
   \]

   b. (3 pts) Write an OCaml expression with the following type

   \[
   (\text{float list} \to \text{float} \to 'a) \to 'a \quad \text{Code } = \text{One possible answer:}
   \]

   \[
   \text{fun } x \to x \ [1.0] \ 3.0;
   \]

   c. (3 pts) Give the value of the following OCaml expression. If an error exists, describe the error. The function fold is given below.

   \[
   \text{fold } ((\text{fun } x \ y \ z \to x + (y \ast z)) \ 2) \ 1 \ [1; \ 2; \ 3]; \quad \text{Value } = 26
   \]

2. (8 pts) OCaml Programming

Using either map or fold and an anonymous function, write a curried function called \texttt{divisible} which when given a number \( n \) and a list of ints \( \text{lst} \), returns a list of all elements of \( \text{lst} \) that are divisible by \( n \) (maintaining their relative ordering). You are allowed to use \texttt{List.rev} (reverses a list) and the (curried) map and fold functions provided, but no other OCaml library functions. \textbf{Hint:} \( x \) is divisible by \( y \) iff \( (x \mod y = 0) \) is true.

\[
\begin{align*}
\text{let rec map } f \ l & = \text{match } l \text{ with} \\
& \quad [\] \to [] \\
& \quad | (h::t) \to (f \ h)::(\text{map } f \ t)
\end{align*}
\]

\[
\begin{align*}
\text{let rec fold } f \ a \ l & = \text{match } l \text{ with} \\
& \quad [\] \to a \\
& \quad | (h::t) \to \text{fold } f \ (f \ a \ h) \ t
\end{align*}
\]

Example:

\[
\begin{align*}
\text{divisible } 4 \ [3;16;24] & \quad \text{// returns } [16; \ 24] \\\n\text{divisible } 3 \ [4;1;11] & \quad \text{// returns } [] \\
\text{divisible } 3 \ [ ] & \quad \text{// returns } []
\end{align*}
\]

One possible solution:

\[
\begin{align*}
\text{let divisible } v \ \text{lst} & = \text{List.rev} \\
(\text{fold } (\text{fun } a \ h \to \text{if } (h \mod v = 0) \text{ then } (h::a) \text{ else } a) \ [ ] \ \text{lst})
\end{align*}
\]
3. (4 pts) Context Free Grammars

Consider the following grammar:

\[ S \rightarrow aSc \mid b \mid \text{epsilon} \]

a. (2 pts) Describe the set of strings accepted by this grammar.

Strings of a’s, followed by 0 or 1 b’s, followed by c’s, where the number of a’s and c’s is the same (and may be 0). The empty string is also part of the language.

OR

String of N a’s followed by N c’s, where N ≥ 0. The a’s and c’s can possibly separated by a single b. Also includes the empty string.

b. (2 pts) Draw a parse tree for the string aabcc.