1. (8 pts) OCaml Types and Type Inference
   a. (3 pts) Give the type of the following OCaml expression
      \[ \text{fun } x \rightarrow [ x \, 1 ] \]
      \[ \text{Type } = \quad \text{(int -> 'a)} ightarrow ('a list) \]
   b. (3 pts) Write an OCaml expression with the following type
      \[ 'a \text{ list } ightarrow 'a \]
      Code = fun (x::y) -> x
            fun x -> match x with h::t -> h
   c. (2 pts) Give the value of the following OCaml expressions. If an error exists, describe the error.
      \[ \text{(fun } x \rightarrow \text{fun } y \rightarrow x+y) \, 6 \, 4 \]
      Value = 10

2. (16 pts) OCaml Programming
   a. (8 pts) Write a curried function \( \text{findKth} \) which when given a number \( k \) and a list \( \text{lst} \) of int (key, value) pairs, returns the \( k \)th value in the list. You may use map or fold if you wish, but it is not required. You may assume \( \text{lst} \) contains at least \( k \) pairs. Example:
      \[ \text{findKth} \, 1 \, [(1,2);(5,9);(9,3)] = 2 \quad \text{// since 2 is 1st value} \]
      \[ \text{findKth} \, 2 \, [(1,2);(5,9);(9,3)] = 9 \quad \text{// since 9 is 2nd value} \]
      \[
      \text{let rec findKth } k \, \text{lst} = \text{match lst with} \\\n      \quad (x,y)::t \rightarrow \text{if } k = 1 \text{ then } y \text{ else } (\text{findKth} \, (k-1) \, t) \]
   b. (8 pts) Using either map or fold and an anonymous function, write a curried function \( \text{findGreaterThan} \) which when given a number \( n \) and a list of ints \( \text{lst} \), returns a list of all elements of \( \text{lst} \) greater than \( n \) (maintaining their relative ordering). You may assume \( (x > y) \) returns true when \( x \) is larger than \( y \). Example:
      \[ \text{findGreaterThan} \, 20 \, [33;18;21;19] = [33;21] \]
      \[ \text{findGreaterThan} \, 65 \, [33;18;21;19] = [] \]
      \[
      \text{let findGreaterThan } v \, \text{lst} = \text{List.rev} \\\n      \quad (\text{fold} \ (\text{fun } a \, h \rightarrow \text{if } (h > v) \text{ then } h::a \text{ else } a) \, []) \, \text{lst} \]

3. (6 pts) Context Free Grammars
   Consider the following grammar:
   \[
   S \rightarrow E+E | E*E \\
   E \rightarrow 0 | 1 | n | (S) \\
   \]
   a. (2 pts) What is the set of strings accepted by this grammar?
      \textbf{Arithmetic expressions involving + and *.}
   b. (4 pts) Provide a \textit{leftmost} derivation of the string \( "(n+1)^n" \) for this grammar.
      \[ S \rightarrow E*E \rightarrow (S)*E \rightarrow (E+E)*E \rightarrow (n+E)*E \rightarrow (n+1)*E \rightarrow (n+1)^n \]