Please do not turn the page until everyone is ready.

Rules:

• The exam is closed-book, closed-note, except for one side of one 8.5x11in piece of paper.
• Please stop promptly at 1:20.
• You can rip apart the pages, but please write your name on each page.
• There are a total of 60 points, distributed unevenly among 5 questions (each with multiple parts).
• When writing code, style matters, but don’t worry about indentation.

Advice:

• Read questions carefully. Understand a question before you start writing.
• Write down thoughts and intermediate steps so you can get partial credit.
• The questions are not necessarily in order of difficulty. Skip around.
• If you have questions, ask.
• Relax. You are here to learn.
1. Consider this datatype for non-empty lists (but not built-in ML lists) of integers:

\[ \text{datatype } t = \text{One of int} \mid \text{More of int} \times t \]

(a) (4 points) Write an ML function \texttt{length} that takes \texttt{t} and returns how many \texttt{int} values are in the \texttt{t}. Your solution must \textit{not} be tail-recursive.

\textbf{Solution:}

\begin{verbatim}
fun length lst =
  case lst of
    One _ => 1
  | More(_,m) => 1 + length m
\end{verbatim}

(b) (6 points) Write an ML function \texttt{rev_map} that takes 3 arguments (as a tuple): (1) A function \texttt{f} from integers to integers, (2) a \texttt{t} called \texttt{acc}, and (3) a \texttt{t} called \texttt{lst}. The function should return a \texttt{t} that is the result of reversing \texttt{lst}, applying \texttt{f} to every \texttt{int} to the reversed list, and appending that result to \texttt{acc}. For example,

\texttt{rev_map ((fn x => x+1), (More (0, One(1))), (More(3, More(4, One(5)))))}

should evaluate to:

\texttt{More(6, More (5, More(4, More(0, One(1))))})

Implement \texttt{rev_map} as a \textit{tail-recursive} function that uses no helper functions.

\textbf{Solution:}

\begin{verbatim}
fun rev_map (f,acc,lst) =
  case lst of
    One i => More(f i, acc)
  | More(i,tl) => rev_map(f,More(f i, acc),tl)
\end{verbatim}

(c) (3 points) What is the type of \texttt{rev_map}?

\textbf{Solution:}

\[ (\text{int}\to\text{int}) \times \text{t} \times \text{t} \to \text{t} \]
2. For each of the following programs, give the value that \texttt{ans} is bound to after evaluation. Underlining is just to help you see the differences between problems.

(a) (3 points)
\[
\text{fun } f \ x = \\
\quad \text{let } \texttt{val } x = x + 1 \\
\quad \text{val } y = x + 1 \\
\quad \text{in} \\
\quad \quad y + 1 \\
\quad \text{end} \\
\quad \text{val } x = 1 \\
\quad \text{val } \texttt{ans} = f \ x
\]
Solution:
4

(b) (3 points)
\[
\text{fun } f \ x = \\
\quad \text{let } \texttt{val } y = x + 1 \\
\quad \texttt{val } x = x + 1 \\
\quad \text{in} \\
\quad \quad y + 1 \\
\quad \text{end} \\
\quad \text{val } x = 1 \\
\quad \text{val } \texttt{ans} = f \ x
\]
Solution:
3

(c) (3 points)
\[
\text{fun } f \ (x,y) = \\
\quad \text{if } x=10 \\
\quad \quad \text{then } (\text{fn } x \Rightarrow x + y) \\
\quad \quad \text{else } (\text{fn } x \Rightarrow x - y) \\
\quad \text{val } x = f(3,4) \\
\quad \text{val } \texttt{ans} = x \ 10
\]
Solution:
6

(d) (4 points)
\[
\text{fun } f \ (x,y) = \\
\quad (\text{fn } x \Rightarrow \text{if } x=10 \\
\quad \quad \text{then } x + y \\
\quad \quad \text{else } x - y) \\
\quad \text{val } x = f(3,4) \\
\quad \text{val } \texttt{ans} = x \ 10
\]
Solution:
14
3. Consider this ML function:

```ml
fun someFun (f,g,start,stop) =  
  let fun loop n =  
    (n <= stop) andalso ((f (g n)) orelse (loop (n + 1)))  
  in  
    loop start  
  end
```

(a) (5 points) Fill in the blanks to give the type of `someFun`.
Hint: The solution has one type variable, which appears twice.

```
(_______________ *  
  _______________*  
  _______________*  
  _______________ ) -> _______________
```

Solution:

```
('a->bool)*(int->'a)*int*int->bool
```

(b) (7 points) What does `someFun` compute? (Describe what it computes from a caller’s perspective, not how `someFun` works. Start your answer with

“someFun(f,g,start,stop) evaluates to ______ if and only if ...”.)

Solution:

`someFun(f,g,start,stop)` evaluates to `true` if and only if there exists a number `n` between `start` and `stop` (inclusive) such that applying `f` composed with `g` to `n` evaluates to `true`.

(c) (2 points) Fill in the blank so evaluating this programs produces `true`:

```ml
val x = ______  
someFun((fn z => z = 6), (fn y => x * y), x, (x + 1))
```

Solution:

2
4. Each pair of expressions below is not totally contextually equivalent. Briefly explain why. (Underlining just emphasizes differences.)

(a) (3 points)
   \[
   \text{let val } x = 0 \text{ in } x \text{ end and let val } x = (f \ 3) - (f \ 3) \text{ in } x \text{ end}
   \]

(b) (4 points)
   \[
   (\text{fn x:int } => \text{ fn z:int } => x - y) \ y \text{ and } (\text{fn x:int } => \text{ fn y:int } => x - y) \ y
   \]

Solution:

(a) Evaluating \(f \ 3\) might have an effect (infinite-loop, exception, I/O, etc.)

(b) The first expression evaluates to a function that always returns 0. The second expression evaluates to a function that subtracts its argument from \(y\). For example, in an environment where \(y\) maps to 3, the first expression is equivalent to \((\text{fn z } => 0)\) and the second expression is equivalent to \((\text{fn y } => 3 - y)\).
5. Consider this ML structure definition:

```ml
structure M :> MSIG =
struct
  type one_or_two = bool * int * int
  fun abs_val i = if i < 0 then ~i else i
  fun mkOne i = (false, (abs_val i), ~1)
  fun mkTwo (i, j) = (true, (abs_val i), (abs_val j))
  fun last (x : one_or_two) = if #1 x then #3 x else #2 x
end
```

(a) **(4 points)** Why is the definition of type `one_or_two` bad style? Suggest a different way to program this idea that uses ML's features more appropriately. (Hint: We are asking about the type definition. The fact that `last` doesn't use pattern-matching is not the answer.)

(b) For each of the following `MSIG` definitions, determine if a client can make a call to `last` evaluate to a negative number. **Explain briefly.**

i. **(3 points)**

```ml
signature MSIG =
sig
  type one_or_two = bool * int * int
  val mkOne : int -> one_or_two
  val mkTwo : int * int -> one_or_two
end
```

ii. **(3 points)**

```ml
signature MSIG =
sig
  type one_or_two = bool * int * int
  val mkTwo : int * int -> one_or_two
  val last : one_or_two -> int
end
```

iii. **(3 points)**

```ml
signature MSIG =
sig
  type one_or_two;
  val mkOne : int -> one_or_two
  val mkTwo : int * int -> one_or_two
  val last : one_or_two -> int
end
```

**Solution:**

(a) Because `one_or_two` is a "one of" type, so a **datatype** binding, e.g., `datatype one_or_two = One of int | Two of int * int` is better style.

(b) i. No, `last` is not in the signature; no client can cause `last` to be called with any arguments.
   ii. Yes, a client can just do `M.last(true, 0, ~1)`.
   iii. No, `one_or_two` is abstract, so any value of this type was produced by `mkOne` or `mkTwo`. Looking at these functions, that means either the `bool` is `false` and the second `int` is non-negative, or the third `int` is non-negative. In either case, `last` evaluates to a non-negative number.