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:

   ```ml
datatype t = One of int | More of int * t
```

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

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

   ```ml
   rev_map ((fn x => x+1), (More (0, One(1))), (More (3, More (4, One (5)))))
   ```

   should evaluate to:

   ```ml
   More (6, More (5, More (4, More (0, One (1)))))
   ```

   Implement `rev_map` as a *tail-recursive* function that uses no helper functions.

(c) **(3 points)** What is the type of `rev_map`?
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) \textbf{(3 points)}

\begin{verbatim}
fun f x = 
  let val x = x + 1 
  val y = x + 1 
  in 
  y + 1 
  end 
val x = 1 
val ans = f x 
\end{verbatim}

(b) \textbf{(3 points)}

\begin{verbatim}
fun f x = 
  let val y = x + 1 
  val x = x + 1 
  in 
  y + 1 
  end 
val x = 1 
val ans = f x 
\end{verbatim}

(c) \textbf{(3 points)}

\begin{verbatim}
fun f (x,y) = 
  if x=10 
  then (fn x => x + y) 
  else (fn x => x - y) 
val x = f(3,4) 
val ans = x 10 
\end{verbatim}

(d) \textbf{(4 points)}

\begin{verbatim}
fun f (x,y) = 
  (fn x => if x=10 
  then x + y 
  else x - y) 
val x = f(3,4) 
val ans = x 10 
\end{verbatim}
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.

```
(______________ * 
 ______________ * 
 ______________ * 
 ______________ ) => ______________
```

(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 ...").

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

```
val x = ____
val someFun((fn z => z = 6), (fn y => x * y), x, (x + 1))
```
4. Each pair of expressions below is *not* totally contextually equivalent. Briefly explain why. (Underlining just emphasizes differences.)

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

(b) (4 points)
\[
\begin{align*}
(\text{fn } x: \text{int} \Rightarrow \text{fn } z: \text{int} \Rightarrow x - y) \ y & \text{ and } (\text{fn } x: \text{int} \Rightarrow \text{fn } y: \text{int} \Rightarrow x - y) \ y
\end{align*}
\]
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
```