There are 5 problems on this exam. You have one and a half hours for the exam. This is a closed-book examination; you may not use outside materials.

Name: ____________________________

Net ID: ____________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. Short Questions [20 pts]  (parts a–d)

(a) [5 pts] Consider the following function definition: \( \text{fun } f \ x \ y = y \ x \ x \)
Pick the appropriate type for \( f \) from the list below:

A. \( 'a \rightarrow ('a \rightarrow 'a \rightarrow 'b) \rightarrow 'b \)
B. \( 'a \rightarrow ('a \rightarrow 'b) \rightarrow ('a \rightarrow 'b) \)
C. \( 'a \rightarrow ('a \rightarrow 'b) \rightarrow 'b \)
D. \( 'a \ast ('a \rightarrow 'b) \rightarrow ('b \rightarrow 'a) \)

The type of \( f \) is:

(b) [5 pts] Show the evaluation of the expression below using the substitution model:

\[
\text{let fun foo (x:int) (y:int) : int = x * y} \\
\text{fun bar (x:int->int, y:int) : int = x y} \\
in \\
\text{bar(foo 2, 3)} \\
end \\
\]

->

(c) [5 pts] Which of the following equalities are true:

(A) \( []::x = [x] \)  \hspace{2cm} (D) \( x::[] = x \)
(B) \( []::x = [[] , [x]] \)  \hspace{2cm} (E) \( x::[] = [x] \)
(C) \( []::x = [[] , x] \)  \hspace{2cm} (F) \( x::[] = [x, []] \)

True equalities:

(d) [5 pts] Order the following functions in ascending order with respect to their order of growth: \( n^2 \), \( n \ log(n^2) \), \( 2n^2 + n \), \( (log n)^2 \).

Answer:
2. List Manipulations [24 pts] (parts a–b)

(a) [12 pts] Consider a function remthree that removes every third element from a list. For instance:

\[
\text{remthree } [1, 2, 3, 4, 5, 6, 7, 8] = [2, 3, 5, 6, 8]
\]

Use pattern matching to implement function remthree.

(b) [12 pts] Consider a function that removes all consecutive duplicates from a list. For instance:

\[
\text{remdups } [1, 2, 2, 3, 3, 3, 1, 1] = [1, 2, 3, 1]
\]

The function can be implemented using fold, as follows:

\[
\text{val remdups : int list -> int list = foldr func []}
\]

where func is an appropriate function and foldr has the standard definition:

\[
\text{fun foldr(f: 'a * 'b -> 'b) (v:'b) (l:'a list) =}
\]

\[
\text{case l of [] => v}
\]

\[
\text{| h::t => f(h, foldr f v t)}
\]

Write the code for function func. Make sure you indicate the types of the argument(s) and return value in the function definition.
3. Data Abstraction [24 pts] (parts a–d)

The following defines a data abstraction interface and a partial implementation of this interface:

signature INTSET = sig
(* A "set" is a set of integer numbers. *
 * Examples: \{3, 1, 2\}, \{-1, 1\}, \{} *)
type set

(* empty() returns an empty set. *)
val empty : unit -> set

(* add(s,n) adds element n to set s. *)
val add: set * int -> set

(* ... *)
val oper: set * set -> set

(* size(s) is the number of elements in the set s. *)
val size: set -> int
end

structure Set : INTSET = struct

  type set = int list

  fun empty() = []
  fun add(s, n) = raise Fail "unimplemented!"
  fun size l = List.length l

  fun oper(s1, s2) =
    case (s1, s2) of
    ( ([],_), (_, []) ) => []
    | (h1::t1,h2::t2) => case Int.compare(h1,h2) of
      EQUAL => h1::oper(t1,t2)
      | LESS => oper(t1,s2)
      | GREATER => oper(s1,t2)
    end

(a) [4 pts] Write an appropriate specification for function oper, specifying the operation that it implements.
(b) [6 pts] Write an appropriate abstraction function and representation invariant for this implementation.

(c) [4 pts] Explain why the invariant must hold for this implementation.

(d) [10 pts] Write the appropriate code for function add.
4. **Induction** [16 pts]
   Consider a tree data type and two functions, `edges` and `nodes`, defined for this type:

   ```plaintext
   datatype tree = Nil | Node of int * tree * tree
   
   fun nodes(l: tree): int = 
   case l of Nil => 0
   | Node(_, l, r) => 1 + nodes(l) + nodes(r)
   
   fun edges(l: tree): int = 
   case l of Nil => ~1
   | Node(_, l, r) => 2 + edges(l) + edges(r)
   ```

   Use induction to prove that \(\text{nodes}(t) = \text{edges}(t) + 1\) for all trees. Make sure you clearly show all of the steps in your proof, and state the kind of induction you’re using.
5. Complexity [16 pts] (parts a–b)

Consider the following implementations for an exponentiation function that raises a
count $x$ to a power $n$:

\[
\begin{align*}
(* \text{ Returns } x^n. \text{ Requires } n \geq 0. *)
\text{fun exp1 (x:int, n:int) =}
\begin{cases}
(0, _) & \Rightarrow 1 \\
(_, 0) & \Rightarrow x * x * \text{exp1}(x, n - 2) \\
(_, _) & \Rightarrow x * \text{exp1}(x, n - 1)
\end{cases}
\end{align*}
\]

\[
\begin{align*}
(* \text{ Returns } x^n. \text{ Requires } n \geq 0. *)
\text{fun exp2 (x:int, n:int) =}
\begin{cases}
(0, _) & \Rightarrow 1 \\
(_, 0) & \Rightarrow \text{exp2} (x * x, n \text{ div } 2) \\
(_, _) & \Rightarrow x * \text{exp2}(x, n - 1)
\end{cases}
\end{align*}
\]

(a) [10 pts] For each of the functions above, derive recurrence relations describing
their running time $T(n)$, where $n$ is the second argument of each function.
(b) [6 pts] State the asymptotic complexity of each of the two functions with respect to \( n \), their second argument. You don’t need to prove your results.