**Practice Midterm 1**  
*(October 8, 2017)*

This exam is closed-book, notes, and technology.

You have 50 minutes to complete this exam.  
Please do not open the test until the instructor says time has begun.  
Please only write on the front of pages on the exam.  
Please rebind your exam with a paperclip if you decide to detach the pages.

Please stop writing once the instructor has called time.  
Failure to stop writing will result in a zero on the exam.

Remember you are here to learn.  
Relax and think of this as yet another learning experience.

Good luck, have fun!

Your Name: __________________________________________________________________________________

<table>
<thead>
<tr>
<th></th>
<th>/ 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>/ 100</strong></td>
</tr>
</tbody>
</table>
Problem 1:  Symbolic (20 points)

(a) Consider the following OCaml program:

```ocaml
let foo (x:int) (y:int) : int =
  let z = x - y in
  if z < 0 then
    let a = z * z in a + x + y
  else
    let a = x * x in a - z - y
```

Give the complete step-by-step evaluation of the following expression: `foo (-5) 3`. Please write down the complete expression at every step of your derivation.

(b) Consider the following OCaml program:

```ocaml
type nat = O | S of nat

let rec bar (n1:int) (n2:int) : int =
  match n1 with
  | O   -> n2
  | S n1' -> S (bar n1' n2)
```

Give the complete step-by-step evaluation of the following expression: `bar (S (S O)) (S (S O))`. You may elide the branches of `match` expressions in your derivation, e.g., `match (S (S O)) with ....`. 
Problem 2: Logic (20 points)

(a) Show that the logical equivalence:

\[(A \land B) \rightarrow \neg B = B \rightarrow (\neg A \lor \neg B)\]

holds by giving a complete truth table for both propositions. Your truth table should include an entry for every sub-expression of each proposition.

(b) Define the following predicates:

\[P(x, y) = x \text{ is in the same class as } y\]
\[Q(x) = x \text{ is in CSC 208}\]

Translate the following formal proposition into an English sentence: \(\forall x. \exists y. P(x, y)\):

Translate the following informal proposition into a formal proposition using \(P\) and \(Q\) as defined above: “A student in CSC 208 has a class with everyone else in the college”:

(c) Give a complete natural deduction proof that the following proposition holds, i.e., it's a tautology:

\[A \lor \neg B \rightarrow (\neg B \rightarrow A) \rightarrow A.\]
Problem 3: Induction (20 points)

Consider the following recursive OCaml functions over lists:

(* Returns true iff all the booleans contained in l are true with list_and [] = true. *)
let rec list_and (l: bool list) : bool = (* ... *)

(* Returns the maximal element of the given list. pre: l != [] *)
let rec max (l: int list) : int = (* ... *)

(* returns the product of the elements of the list with product [] = 1. *)
let rec product (l: int list) : int = (* ... *)

(a) For each function, write down a proposition that implies the (partial) correctness of the function.

(b) For each of the propositions stated above, write down the induction hypothesis that we may assume in the inductive case of their respective proofs.

(c) Choose one of the functions and implement it below.

(d) For the function that you chose, give in a sentence or two the high-level intuition as to why the proposition you stated in (b) for that function holds. Recall that such a “proof” should proceed via case analysis, but does not need to go through the details of symbolic evaluation.
Problem 4: Programming (20 points)

(a) Implement the contains function over lists. contains takes a value (of type 'a) and a list (of type 'a list) and returns true iff the list contains that value.

(b) Using contains, implement the union function over lists. union behaves like list concatenation, except duplicates of the lists are removed. For example, union [1; 2; 3] [2; 3; 4] = [1; 2; 3; 4].
Problem 5: Proof (20 points)

Consider the following OCaml implementation of the `pairwise_swap` function:

```ocaml
(* Returns a new list that is the result of swapping the position of * consecutive elements of the list. If the list has odd length, then * the final element of the list is unmodified. For example: * pairwise_swap [1; 2; 3; 4; 5] = [2; 1; 4; 3; 5] *)
let rec pairwise_swap (l:'a list) : 'a list =
  match l with
  | [] -> []
  | [x] -> [x]
  | x :: y :: xs -> y :: x :: pairwise_swap xs
```

And formally prove the following claims over this function:

**Claim 1 (Identity).** $\exists l. \text{pairwise}_\text{swap} \ l = l$.

**Claim 2 (Involution).** $\forall l. \text{pairwise}_\text{swap} \ \text{pairwise}_\text{swap} \ l = l$.
(This page is scratch paper.)
(This one, too, since we have an even number of pages!)