CS 477, Fall 2016
Midterm Exam

NAME:

NETID (e.g., rloyd67):

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possible</td>
<td>30</td>
<td>20</td>
<td>15</td>
<td>20</td>
<td>85</td>
</tr>
<tr>
<td>Score</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You have 70 minutes to finish the exam.

We will be checking photo ID’s during the exam. Have your ID handy.

Turn in your exam at the front when you are done.
Problem 1: Multiple choice (10x3pts = 30 points)

In this question, there are some problems where multiple choices could be marked, and some where a single choice needs to be marked. Read the questions carefully.

(a) The Hoare triple \( \{ x = 1 \land y > 1 \} \text{ while } (x<y) \{ x:=x+1; \} \{ x = y \} \) is valid over unbounded integers.

Valid \( \checkmark \) Not valid  

(b) A Hoare triple \( \{ P \} S \{ Q \} \) is valid if:

(1) If \( S \) starts in a state satisfying \( P \), then it will end finally in a state satisfying \( Q \)  
(2) If \( S \) starts in a state satisfying \( P \), then it may end up in a state satisfying \( Q \)  
(3) If \( S \) ends in a state satisfying \( Q \), then the state it began from must be in \( P \)  
(4) If \( S \) starts in a state satisfying \( P \), then if it finishes, it will be in a state satisfying \( Q \) \( \checkmark \)  
(5) If \( S \) ends in a state not satisfying \( Q \), then the state it began from must not have satisfied \( P \) \( \checkmark \)

(c) In the theory of uninterpreted functions \( (\forall x. (f(x) = f(f(x)) ) \Rightarrow \forall y. (g(f(f(f(y)))) = g(f(y))) \) is valid.

Valid \( \checkmark \) Not valid  
Satisfiable \( \checkmark \) Unsatisfiable
(d) The (first-order) theory of (quantified) uninterpreted functions is decidable.  

True  False  

(e) There is a sound and complete finite axiomatization of the first-order theory of integers with addition and multiplication.  

True  False  

(f) The Hoare triple \( \{ x = 1 \} \left\{ \text{while} \ (x>0) \ \{ x:=x+1; \} \right\} \{ x = 5 \} \) is
(where \( x \) is a variable over unbounded integers)

Valid  Not valid  

(g) Consider the Hoare triple \( \{ P \} \left\{ \text{while} \ (x > y) \ \{ x:=x-1; \ y:=y+1; \} \right\} \{ x = y \} \) is
(where \( x, y \) are variables over unbounded integers)

Which of the following formulae for \( P \) make the above triple valid:
(Choose all options that apply.)

\[
\begin{align*}
x = 15 \land y = 0 & \quad \text{false} \\
x = 241 \land y = 0 & \quad \text{true} \\
\text{even}(x) \land y = 0 & \quad \text{false} \\
\text{true} & \\
\end{align*}
\]
(h) Consider a program with only Booleans and pure integers as datatypes, where there are no function-calls (only while-loops), and where expressions in assignments and guards only involve either Boolean operators or addition. Given such programs annotated with pre- and post-conditions, and where every loop is provided with an invariant, where all these annotations are quantifier free and use only the theory of arithmetic with addition, verifying whether the program respects these annotations is

Decidable  ✓  Undecidable  

(i) The validity problem for general first-order logic is

Decidable  
Recursively enumerable  ✓

Undecidable  ✓

(j) Professor Moriarty wants to check whether a program $P$ terminates (starting from any state) by proving/disproving the Hoare triple $\{true\}P\{false\}$
Which of the following are true?
(check only one option)

If $\{true\}P\{false\}$ is valid, then $P$ terminates from any state  
If $\{true\}P\{false\}$ is valid, then $P$ does not terminate from any state  ✓
If $\{true\}P\{false\}$ is valid, then $P$ does not terminate from some state  ✓
If $\{true\}P\{false\}$ is valid, then $P$ terminates from some state  

3
Problem 2: Short problems \((5+5+5+5=20 \text{ points})\)

(a) Consider the FOL formula \(\forall x. (f(g(x)) = g(f(x))\). Construct a model for this formula with two elements.

\[
\text{Universe: } U = \{a\} \\
\begin{align*}
  f(a) &= a \\
  g(a) &= a
\end{align*}
\]

Clearly \(\forall x \in U, f(g(x)) = f(a) = a\) and \(g(f(x)) = g(a) = a\).

(b) Consider the following program:

\[
\begin{align*}
  x &:= y+9; \\
  \text{if } (x>5) \text{ then } \\
  &\quad \text{do} \\
  &\qquad x := 2 \times x; \\
  &\qquad \text{if } (x>120) \text{ then} \\
  &\qquad \quad x := x+1; \\
  &\text{else} \\
  &\quad \text{opensocket(); } // \text{ We need to reach here}
\end{align*}
\]

In the above, \(x\) and \(y\) are integer variables (over unbounded integers). We would like to find a test input value for \(y\) that will drive the program down to the place where \texttt{opensocket} is called. Write down the path condition for this—i.e., write down a logical formula \(\varphi\) without quantifiers in which \(y\) is one of the free variables such that the set of values for \(y\) that will drive the program down the required path is precisely the set of values of \(y\) in satisfying models of the formula. Moreover, you should derive this condition systematically from the path (do not just write down the final formula—show how you derive it systematically from the path!).

Does your formula lie in a theory that has a decidable satisfiability problem?

Path leading to \texttt{opensocket}: 
\[
\begin{align*}
  x &:= y+9 \\
  x &> 5 \\
  x &:= 2 \times x \\
  x &\leq 120
\end{align*}
\]

Path condition: 
\[
\begin{align*}
  x &= y+9 \land x > 5 \land x' = 2x \land x' \leq 120
\end{align*}
\]

Alternatively, we can do weak path:
\[
\begin{align*}
  2x \leq 120 \land x > 5 \land x = y+9 \land x' = 2x \land x' \leq 120
\end{align*}
\]

We can also quantify \(x, x'\) out. 
\[
\exists x \forall x' (x = y+9 \land x > 5 \land x' = 2x \land x' \leq 120)
\]

\[...\text{Use space on next page too...}\]
Or substitute values for \( x \) and \( x' \):

\[
y + 9 \geq 5 \quad \land \quad 2(y + 9) \leq 120
\]

All the above fall in theory of arithmetic with
addition (linear arithmetic/Presburger arithmetic)
which is decidable.

(c) What is the while proof rule in Hoare logic? Write down the rule and explain what it means and why it is a sound rule.

\[
\{ P \land B \} \; S \; \{ P \} \\
\{ P \} \; \text{while} \; B \; \text{do} \; S ; \od \; \{ P \lor B \}
\]

It says that if \( P \) is invariant across \( S \) provided \( B \) is true before execution \( S \),
then we can conclude that \( P \) will hold after a "while \( B \) do \( S \)"
and further that \( B \) will hold in the end.

Soundness: If \( \{ P \land B \} \; S ; \{ P \} \) holds, then we know that
as long as \( B \) holds, \( P \) will continue to hold. Hence when
\( \text{while loop exits} \), \( P \) will still hold too.

\[\text{at some rule we are loop.}\]
When the loop exits, P will still be in train B will now be in the middle since we are not in the loop.
(d) Give invariants and a ranking function that proves that the following program terminates. Argue why the ranking function maps to a well-ordered set, and why it decreases across the loop, in words.

```c
function f(int x)
  requires x>0
  { int y:=0;
    while (y<x) {
      y:=y+1;
      x:=x-2;
    }
  }
```

First, $x \geq -1 \land y \geq 0$ is an invariant:

- it holds in the beginning since $x>0$ and $y=0$
- it's preserved across the loop:
  - if $x \geq -1 \land y \geq 0$ and $y < x$ then $x > 0$
  - and hence $x' = x - 2 \geq -1$ and $y' = y + 1 \geq 0$.

So the ranking function always maps each state in the invariant to a value $\geq 0$.

Also, since $x$ decreases by 2 across the loop body, the value of $x+1$ also decreases across the loop body.

Hence the rank goes down in an iteration of the loop.

Hence the program has to terminate on all inputs satisfying the pre-condition.

Note: $x-y+2$ is also a valid ranking function with invariant $x-y+2 \geq 0 \land y \geq 0$. 

6
Problem 3: Hoare-style proof (15 points)

For the program below write down the loop-invariant and prove the program correct in Hoare logic. You need to write a proof where every rule follows the Hoare logic rules. However, once you get to a node where the proof involves only logic over arithmetic (and no program), you can stop there (but ensure that it is a valid statement in arithmetic!).

\{x = 2\} 
while(x \leq y) do
  x := x + x;
\{\forall i. x \neq 3\} 

1. \{even(x) \land x \geq 2\} \ x := x + x \ \\text{even(x)} \land x \geq 2 \text{ by Assignment Axiom}

2. \{e\text{ven}(x) \land x \geq 2 \land x \leq y\} \ x := x + x \ \text{even(x)} \land x \geq 2 \text{ by (1) and Rule of Consequence}

3. \{even(x) \land x \geq 2\} \ \text{while (x \leq y) do } x := x + x; \ \{\text{even(x)} \land x \geq 2\} \text{ by (2) and Loop Rule}

4. \{x = 2\} \ \text{while (x \leq y) do } x := x + x \ \{\forall i. x \neq 3\} \text{ by Rule of Consequence using (3).}

\_{QED.}

Note: Of course, there are many loop invariants that can prove the program correct.
Problem 4: Verification conditions (20 points)

Consider the following program:

\[
\{n \geq 0\} \\
i := 0 \\
s := 0 \\
\textbf{while}(i \neq n) \textbf{do} \\	\quad i := i + 1; \\	\quad s := s + (2 \times i - 1); \\
\{s = n^2\}
\]

1. Write down an adequate inductive loop invariant that proves the program against its postcondition.

2. Write down all the basic paths, and for each basic path, derive the \textit{weakest pre-conditions} of the post-condition for the basic path, and using it, derive the verification condition. Ensure that your verification conditions are valid.

1. Adequate loop invariant: \(s = i^2\)

2. \(BP1: \{n \geq 0\} i := 0; s := 0 \{s = i^2\}\)

\[
\text{wp}(s = i^2, i := 0; s := 0) = \text{wp}(\text{wp}(s = i^2, s := 0), i := 0) \\
= \text{wp}(0 = i^2, i := 0) = (0 = 0^2)
\]

\(VC: n \geq 0 \Rightarrow 0 = 0^2 \quad \text{(valid)}\)

\(BP2: \{s = i^2\} \text{ assume } i \neq n \; ; i := i + 1; \; s := s + (2 \times i - 1) \{s = i^2\}\)

\[
\text{wp}(s = i^2, \text{ assume } i \neq n \; ; i := i + 1; \; s := s + (2 \times i - 1)) \\
= \text{wp}(\text{wp}(s = i^2, s := s + (2 \times i - 1)), \text{ assume } i \neq n \; ; i := i + 1)) \\
= \text{wp}(s + (2 \times i - 1) = i^2, \text{ assume } i \neq n \; ; i := i + 1) \\
= \text{wp}(\text{wp}(s + 2(i - 1) = i^2, i := i + 1), \text{ assume } i \neq n) \\
= \text{wp}(s + 2(i + 1) - 1 = (i + 1)^2, \text{ assume } i \neq n) \\
= i \neq n \Rightarrow s + 2(i + 1) - 1 = (i + 1)^2
\]
VC: $S = i^2 \Rightarrow \left( i \neq n \Rightarrow s + 2(i+1) - 1 = (i+1)^2 \right)$  

(BP3): $\{ S = i^2 \} \quad \text{assume} \quad \neg (i \neq n) \quad \{ S = i^2 \}$

$\text{wp}(S = n^2, \text{assume} \quad \neg (i \neq n))$

$= \neg (i \neq n) \Rightarrow S = n^2$

VC: $S = i^2 \Rightarrow \left( \neg (i \neq n) \Rightarrow S = n^2 \right)$ (valid).