Practice Test
CSCI 2600 Principles of Software

- DO NOT OPEN THIS EXAM UNTIL TOLD TO DO SO!
- READ THROUGH THE ENTIRE EXAM BEFORE STARTING TO WORK.
- YOU ARE ALLOWED ONLY 2 “CHEAT” PAGES. NO OTHER MATERIAL IS ALLOWED.

This exam is worth 100 points.

Make sure you have 8 pages counting this one. There are 3 parts, each including multiple questions. If you need more room for an answer than is provided, please use the back of the page and indicate that you have done so. If you re-do a question, please make clear what is your final answer.

Be clear and brief in your explanations—rambling and lengthy answers will be penalized. All questions have short answers.

The following is for the use of graders

1. ________/50
2. ________/30
3. ________/20

TOTAL:__________/100
Part I. Reasoning About Code

Note: all variables are integers

1) Order the conditions from strongest to weakest

(a) \( x \geq 10 \) || \( y \) is odd
(b) \( x = 10 \) & \( y \) is odd
(c) \( x \geq 10 \) & \( y \) is odd
(d) \( x = 10 \) & \( y = 15 \)

Answer: _, _, _, _

(a) \( y \) is in interval [0..10]
(b) \( y \) is in [-10..-10]
(c) \( y \) is an odd number in [1..5]
(d) \( y = 3 \)

Answer: _, _, _, _, _

2) Are the following Hoare triples true or false?

a) TRUE / FALSE  (Circle one)

\{ true \}
\z = 0;
if (x > 5)
    \z = x+1;
\{ z > 5 \}

b) TRUE / FALSE

\{ x < -1 || x > 1 \}
if (x < 0)
    \y = x*x;
else
    \y = x-1;
\{ y != 0 \}

3) For each question below, circle TRUE if what is following is a true statement; circle FALSE otherwise.

TRUE / FALSE. If \{ x > 0 \} code Q is true, then \{ x > 1 \} code Q is true.

TRUE / FALSE. If P code \{ x \geq 10 \} is true, then P code \{ x > 10 \} is true.
4) (modified from the UW) Willy Wazoo has another idea about how to simplify the weakest precondition for if-then-else. He thinks that \( \text{wp}(S_1, Q) \lor \text{wp}(S_2, Q) \) will work:

\[
\{ \text{wp}(S_1, Q) \lor \text{wp}(S_2, Q) \}
\]

\[
\text{if } (b) \\
\quad S_1; \\
\text{else} \\
\quad S_2; \\
\} \{ Q \}
\]

Unfortunately, Willy is wrong again. His condition now is too weak. Fill in the blanks below with postcondition \( Q \), \( S_1 \), \( S_2 \) of your choice, then compute \( P \) according to Willy’s formula. Give initial value of \( x \) that satisfies \( P \), but the corresponding final value does not satisfy \( Q \).

\[
\{ P \}
\]

\[
\text{if } (x == 0) \{ \\
\quad x = \\
\}
\]

\[
\text{else} \{ \\
\quad x = \\
\}
\]

\[
\} \{ Q \}
\]
5) Prove that \texttt{power(int x)} produces the correct answer if it terminates.
Recall that \(x/y\) returns the quotient and \(x\%y\) returns the remainder in integer division.

Requires: \(x \neq 0\)
Returns: the highest power \(n\) such that \(2^n\) divides \(x\). E.g., \texttt{power(8)} = 3; \texttt{power(10)} = 1, \texttt{power(144)} = 4, \texttt{power(101101)} = 0

```c
int power(int x) {
    int i = x;
    int n = 0;
    while (i%2 == 0) { // i\%2 gives the remainder mod 2
        i = i/2;
        n = n + 1;
    }
    return n;
}
```

Hint: Think of the number, e.g., 144 as \(9\times2^4\), 4 is the highest power of 2, and 9 mod 2 = 1. Write the postcondition and a suitable loop invariant at the designated places. Then argue \texttt{partial correctness} as we did in class.
Part II. Specifications

1) Satisfying a specification. Given the specifications and implementations below, enter a check mark in the table if an implementation satisfies the corresponding specification.

**Specifications:**
Spec A: “returns: an integer ≥ its argument”
Spec B: “returns: a non-negative integer ≥ its argument”
Spec C: “returns: argument + 1”
Spec D: “returns: argument^2 “
Spec E: “returns: Integer.MAX_VALUE”

**Implementations:**
Code 1: return arg*2;
Code 2: return abs(arg);
Code 3: return arg+5;
Code 4: return arg*arg;
Code 5: return Integer.MAX_VALUE;

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2) Order the specifications below from weakest to strongest

Spec A: requires: integer in interval [0..5]
returns: integer in interval [0..3]
Spec B: requires: true  // true is the weakest condition
returns: false  // false is the strongest condition
Spec C: requires: integer in interval [0..9]
returns: integer in interval [0..2]
Spec D: requires: integer in interval [0..9]
returns: integer in interval [0..1]

Answer: __, __, __, __
3) Below is the Javadoc specification of an `add` method from class `ArrayList`.

```java
public void add(int index, E element)
Inserts the specified element at the specified position in this list. Shifts the element
currently at that position (if any) and any subsequent elements to the right (adds one to
their indices).
Parameters:
   index - index at which the specified element is to be inserted
   element - element to be inserted
Throws:
   IndexOutOfBoundsException - if the index is out of range (index < 0 ||
   index > size())
```

Convert the Javadoc specification into a PoS specification:

**requires:**

**modifies:**

**effects:**

**returns:**

**throws:**

Now, convert your PoS specification into a logical formula:
4) Below is partial implementation of a `remove` method in a collection class:

```java
public void remove(int index) {
    if (index >= size() || index < 0)
        throw new IndexOutOfBoundsException();
    else
        // code to remove element at index from collection
}
```

For each of the specifications below, circle TRUE if the specification is valid for this code.

a) TRUE / FALSE
   
   `requires:` nothing
   `modifies:` this
   `effects:` removes element at index from collection

b) TRUE / FALSE
   
   `requires:` nothing
   `modifies:` this
   `effects:` removes element at index from collection
   `throws:` `IndexOutOfBoundsException` if `index>=size() || index<0`

c) TRUE / FALSE
   
   `requires:` `index<size() && index>=0`
   `modifies:` this
   `effects:` removes element at index from collection

d) TRUE / FALSE
   
   `requires:` `index<size() && index>=0`
   `modifies:` this
   `effects:` removes element at index from collection
   `throws:` `IndexOutOfBoundsException` if `index>=size() || index<0`

e) Which of the above do you think is best? Write a brief explanation.
Part III. ADTs

a) Give two advantages of ADTs. Briefly explain your answer.

b) Classify each method from the Stack ADT as observer, producer, or mutator.

- **empty()**
  - boolean
  - Tests if this stack is empty.

- **peek()**
  - E
  - Looks at the object at the top of this stack without removing it from the stack.

- **pop()**
  - E
  - Removes the object at the top of this stack and returns that object as the value of this function.

- **push(E item)**
  - E
  - Pushes an item onto the top of this stack.

- **search(Object o)**
  - int
  - Returns the 1-based position where an object is on this stack.