NAME______________________ RCS ID_________________

Your RCS ID is the First Part of Your RPI Email
It is not your RIN!

Exam 1
CSCI 2600 Principles of Software
Oct. 4, 2018

- **READ THROUGH THE ENTIRE EXAM BEFORE STARTING TO WORK.**
- **YOU ARE ALLOWED A SINGLE DOUBLE-SIDED SHEET OF NOTES.**
- **NO OTHER MATERIAL IS ALLOWED.**

This exam is worth 100 points.

Make sure you have 10 pages counting this one. 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. Please write neatly. If we cannot read your answer, we cannot grade it.

Unless otherwise specified, all variables are ints. Assume no overflow or underflow conditions occur in any of the code.

The following is for the use by the graders

1. __________/8

2. __________/10

3. __________/8

4. __________/6

5. __________/18

6. __________/18

7. __________/12

8. __________/10

9. __________/10

TOTAL:___________/100

Name: ___________________________

Question 1. (8 points)

Order the conditions from strongest to weakest. Note: All variables are ints.

(a) x = 11 && 1 < y ≤ 10
(b) x = 11 && 1 < y ≤ 100
(c) x is odd && y is odd && x > 0 && y > 0
(d) x == 11 && y == 10

Answer: _ , _ , _ , _

(a) 0 < result < 10 && result is odd
(b) 5 ≤ result ≤ 10
(c) 0 ≤ result ≤ 11
(d) 7 ≤ result ≤ 10

Answer: _ , _ , _ , _

Question 2. (10 pts) For each of the following Hoare Triples, indicate which are valid and which are not necessarily true. If the triple is not valid, explain why or give a counter example. All variables are ints.

a) (VALID / NOT VALID) {true} x++ {x == 1}

b) (VALID / NOT VALID) {x < -1} x = x + 1 {x < 0}

c) (VALID / NOT VALID) {z is even && z > 0} w = z + 3 {w % 2 == 1 && z % 2 == 0}

d) (VALID / NOT VALID) {x == 1 && y > -1} z = x + y {z ≥ 0}

e) (VALID / NOT VALID) {x < y} z = x; x = y; y = z; {x ≥ y}

Question 3. (8 pts) Assume that the following are true. Indicate which Hoare triples are valid. If the triple is not valid, show a counterexample.

{b} code {y}
a ➔ b (a implies b, a is stronger than b)
b ➔ c
c ➔ d
x ➔ y
y ➔ z

a) {c} code {y}

b) {d} code {x}

c) {b} code {z}

d) {a} code {y}

Question 4. (6 pts) Compute the postcondition using **forward** reasoning. Fill in all intermediate conditions at the designated places. Simplify your final postcondition. Assume all variables are ints.

```
{ x > 0 }
y = x +1;
     {____________________________}
z =  y;
     {____________________________}
z = z + 1;
     { ___________________________ }
```

Name: ___________________________

Question 5. (18 pts) Compute the weakest precondition using **backward** reasoning. Fill in all intermediate conditions at the designated places. Simplify your weakest precondition, Assume all variables are ints.

a)

```
P: {____________________________________________________________}

  if (y >= 0) {
             {___________________________________}
          z = x;
  }
  else {
             {___________________________________}
          if(y == -1) {
                    {___________________________________}
                  z = x + 1;
          }
          else {
                    {___________________________________}
                  z = x * x;
          }
  }
  {z == 4}
```

b)

```
P: {_____________________________________}
b = b + 1;
          {_________________________________}
a = b + 3;
{a > 0 && b < 0}
```

c)

```
P:{_____________________________}
y = 4*w –10;
      {_____________________________}
x = 2 * x;
{ x == y }
```

Name: ___________________________

Question 6. (18 pts) TRUE/FALSE.

a) (TRUE/FALSE) If specification A is stronger than specification B, then any implementation that satisfies A satisfies B as well.

b) (TRUE/FALSE) There may exist two logically distinct weakest preconditions A and B for a given bit of Java code. (Logically distinct means that A and B are not just different ways of writing the same logical formula.)

c) (TRUE/FALSE) You can strength a specification by strengthening the precondition.

d) (TRUE/FALSE) The rep invariant maps an object to a Boolean.

e) (TRUE/FALSE) A loop invariant must be true after each statement in a loop.

f) (TRUE/FALSE) Java uses the value model for all variables.

Consider the following code for a search method that performs a linear search on an array:

public static int linearSearch(int[] a, int key)

For each of the following specifications, indicate whether it is a valid specification for this code.

g) (TRUE/FALSE)
    @requires a is not null
    @modifies nothing
    @returns returns i such that a[i] = key if such an i exists and a negative value otherwise

h) (TRUE/FALSE)
    @requires nothing
    @modifies nothing
    @returns returns i such that a[i] = key if such an i exists and a negative value otherwise
    @throws NullPointerException

i) (TRUE/FALSE)
    @requires a is not null and key exists in a
    @modifies nothing
    @returns returns i such that a[i] = key

Question 7 (12 pts)

Prove that the given code below finds the maximum element of an array of ints.

```
// precondition: arr is non-null and non-empty.
int max(int [] arr) {
    int m = arr[0];
    int n = 1;
    while(n < arr.length) {
        if(arr[n] > m) {
            m = arr[n];
        }
        n = n + 1;
    }

    return m;
}
// postcondition: m >= arr[i-1] forall i :: 1 <= i <= arr.length
// in other words m >= every {arr[0], arr[1] … arr[arr.length-1]}
```

a) Show that the loop terminates by identifying a decrementing function that decreases at each iteration. Show that the loop terminates when the decrementing function reaches a minimum.

b) Identify a loop invariant and show that the loop invariant holds prior to executing the loop.

Name: ___________________________

c) Use induction to show correctness of the loop. That is, assume the loop invariant holds after some iteration k and show that it holds after iteration k + 1.

Note, because of the if statement, there are two paths to consider: arr[n] <= m and arr[n] > m.

Name: ________________________

Question 8 (10 points)

Below is a sample Javadoc type specification of a add() method for an ArrayList class.

```
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()`)

a) Convert the Javadoc specification into a PoS specification:

Requires:

Modifies:

Effects:

Returns:

Throws:

b) Now, convert your PoS specification into a logical formula:

Question 9 (10 points)

Consider the following abstract data type.

```
// Overview: A CharSet is a finite mutable set of Characters
// Rep invariant: set has no duplicates

// effects: creates a fresh, empty CharSet
public CharSet ( )

// modifies: this
// effects: thispost = thispre U {c}
public void insert(Character c);

// modifies: this
// effects: this post = this pre - {c}
public void delete(Character c);

// returns: (c ∊ this) i.e. return true if c is an element of CharSet
public Boolean member(Character c);

// returns: cardinality of this
public int size();
```

And the implementation

```
class CharSet {
       private List<Character> elts = new ArrayList<Character>();

       public void insert(Character c) {
              elts.add(c);
       }
       public void delete(Character c) {
              elts.remove(c);
       }
       public Boolean member(Character c) {
              return elts.contains(c);
       }
       public int size() {
              return elts.size();
       }
}
```

There is an error in the implementation that allows violation of the rep invariant.

a) Write a code fragment that will reveal the error.

b) Add code to one of the methods to correct the error.