CSCI2600 Exam 1

Aaron Taylor

TOTAL POINTS

86 / 100

QUESTION 1

1 Question 1 9 / 10

- 0 Correct
- 1 Incorrect x
- 1 Incorrect y
- 1 Incorrect a
- 1 Incorrect b
- 1 Incorrect s
- **1 Incorrect t**

QUESTION 2

2 Question 2 6 / 6

- **0 Correct**
- 1 c is weaker than a
- 1 a is weaker than d
- 1 c is weaker than b
- 1 b is weaker than d
- 1 d is weaker than a

QUESTION 3

3 Question 3 6 / 6

- **0 Correct**
- 1 a) NOT VALID correct
- 1 a) Incorrect counterexample
- 1 b) NOT VALID correct
- 1 b) Incorrect counterexample
- 2 c) VALID correct

QUESTION 4

4 Question 4 6 / 6

- **0 Correct**
- 2 Valid correct
- 1 Not necessarily valid correct
- 1 Counterexample missing
- 2 Valid correct

QUESTION 5

5 Question 5 12 / 12

- **0 correct**
- 1 precondition not properly simplified for a
- 1 precondition not properly simplified for b
- 1 precondition not properly simplified for c
- 2 All steps not correct for a
- 2 All steps not correct for b
- 2 All steps not correct for c
- 4 completely wrong answer for a
- 4 completely wrong answer for b
- 4 completely wrong answer for c

QUESTION 6

6 Question 6 10 / 12

- 0 Correct
- 2 a-1st line
- 2 a-2nd line
- **2 a-3rd line**
- 2 b-first line
- 2 b-2nd line
- 2 c-3rd line
- 3 not completely correct answer for a
- 3 not completely correct answer for b

QUESTION 7

7 Question 7 8 / 12

- **2 Wrong answer**
- 2 Wrong answer
- 2 Wrong answer
- **2 Wrong answer**
- 2 Wrong answer
- 2 Wrong answer
- 0 Full Correct

QUESTION 8

8 Question 8 12 / 12

- 0 correct
- 1 For (i), only a should be checked
- 2 For (i), only a should be checked
- 3 For (i), only a should be checked
- 1 For (ii), a,b,d & e should be checked
- 3 For (ii), a,b,d & e should be checked
- 1 For (iii), only e should be checked
- 2 For (iii), only e should be checked
- 1 For (iv), a,b & e should be checked
- 2 For (iv), a,b & e should be checked
- 1 For (v), a,b & e should be checked
- 2 For (v), a,b & e should be checked
- 3 For (v), a,b & e should be checked

QUESTION 9

9 Question 9 7 / 14

- 0 correct
- 1 Did not mention either n>=0 or p = arr[0]*arr[1]*arr[2]...arr[n-1] in Loop Invariants
- **2 Loop Invariant: n>=0 && p = arr[0]*arr[1]*arr[2]...arr[n-1]**
- 2 Decrementing function: arr.length-n
- 1 Did not mention either n=1/p=arr[0] in base case
- 3 Did not mention Base Case/Incorrect Base Case
- **2 Issues with proof of loop invariant**
- 4 Failure to prove that invariant holds
- 1 Issues with proving decrementing function decreases to zero
- **3 Failure to prove decrementing function decreases to zero**

QUESTION 10

10 Question 10 10 / 10

- **0 Correct**
- 2 Requires: Nothing
- 2 Modifies: The ArrayList
- 2 Effects: Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive. Shifts any succeeding elements to the left (reduces their index).
- 1 Effects: Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive. Shifts any succeeding elements to the left (reduces their index).
- 2 Returns: Nothing
- 2 Throws: IndexOutOfBoundsException
- 10 No answer

NAME Aaron Taylor RCS ID taylo a5

Exam 1
CSCI 2600 Principles of Software
February 16, 2016

- READ THROUGH THE ENTIRE EXAM BEFORE STARTING TO WORK.
- YOU ARE ALLOWED A SINGLE 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.

The following is for the use by the graders

1. ________/10
2. ________/6
3. ________/6
4. ________/6
5. ________/12
6. ________/12
7. ________/12
8. ________/12
9. ________/14
10. ________/10

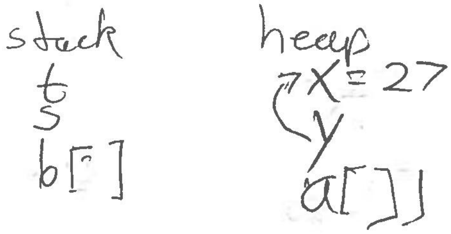
TOTAL:__________**/100**

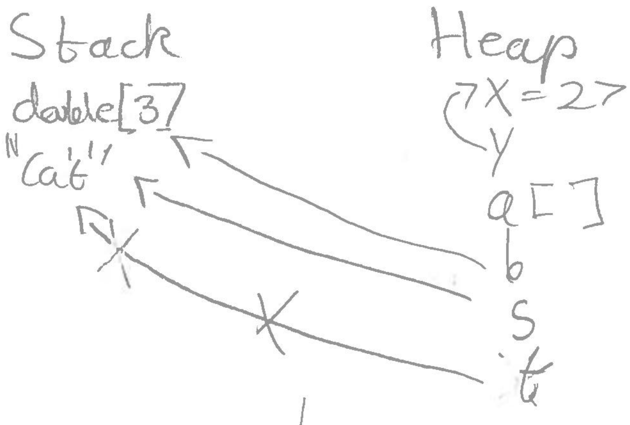
Name: Aaron Taylor

Question 1. (10 pts) Draw the contents of the stack and heap after the execution of the following Java code. If a heap item becomes "garbage", do not erase it, place an x over the reference line that pointed to it.

```
int x = 27;
int y;
double [] a = {1.0,2.0,3.14,4.0};
double [] b = new double[3];
String s = new String("cat");
String t = null;

y = x;
t = s;
s = new String("dog");
t = t + "2";
```





strings are immutable in java, as a result, the reference becomes garbage

Name: Aaron Taylor

Question 2. (6 points)

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

(a) x = 11 && 1 < y =< 10
(b) x is odd and y is even
(c) x is odd && y is prime → can't be compared
(d) x = 11 && y = 10

Answer: D, A, B, C

(a) result = 10
(b) 5 ≤ result ≤ 10
(c) 0 ≤ result ≤ 11
(d) 7 ≤ result ≤ 10

Answer: A, D, B, C

Question 3. (6 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 = 1; x++ {x = 1}

X is being incremented and will not be 1 after

b) (VALID / [✓ NOT VALID]) {x >= -1} x = x + 1 {x > 0}

X can be -1 resolting in x being 0 after the code has finished. This breaks the post-condition.

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

3

1 mod 2 = 1

Name: Aaron Taylor

Question 4. (6 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)
b ➔ c
x ➔ y
y ➔ z

a > b > c
x > y > z

ä > b > y > c
> x > z
y

a) {a} code {y} valid

ā > b and b > y

b) {b} code {x} not valid

b = {x > 1}
x = {x = 2}
y = {x > -1}

Code = x++

b code y => {x ≥ 1} x++ {x > -1} true
b code x => {x ≥ 1} x++ {x = 2} false => can't guarantee x+1 will be 2

c) {b} code {z} valid

b > y and y > z

Name: Aaron Taylor

Question 5. (12 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)                (x <= 1) v (x > 0)
ω P: {(z=x ∧ x > 0) || (z=x+1 ∧ x<-1)}, => {x< -1 or x > 0}
  if (x >=0 ) {
            {        x > 0                     }
        z = x;
  }
  else {
            {        x < - 1                   }
        z = x+1;
  }
  Q: z != 0
```

wp(x ≥ 0 and z != 0 and x ≥ 0

z = x => x != 0

x > 0 and

wp (x > 0 and x < -1 and z != 0)

z != 0 ∧ x <= 0

x+1 != 0

x != -1

wp (z != 0 and x ≥ 0)

z != 0

x+1 != 0

x != 1

wp (x > 0 and z

```
b)             {b < -1}
ω P: {(a=b+3, b > -3) ∧ (b=b++, b < -1)} => {b < -1}
  b = b++;
            {        b+3 > 0                   }
  a = b + 3;
  {a > 0 && b < 0}
```

a > 0

b+3 > 0

b > - 3

b < 0

b+1 < 0

b < -1

$\boxed{b<-1}$ and $\boxed{b>-3}$

wp(b+3 > 0

-3 < b < -1

b+3 >

```
c)
P:{      x > w-10              }
y = w -10;
     {    2x > y               }
x = 2 * x;
{ x > y }
```

y

x > y

2x > y

2x > y

2x > w - 10

2x > w - 10

2x > w - 10 => x > w/2 - 10

x > w - 10

Name: Aaron Taylor

Question 6. (12 pts) Compute the strongest postcondition using **forward** reasoning. Fill in all intermediate conditions at the designated places. Simplify your final postcondition, Assume all variables are ints.

a)

{ x > 0 }

y = x -1;

{ $y \geq 0$ } $\{x > 0 \text{ and } y = x-1\} \Rightarrow x-1 \geq 0 \Rightarrow y \geq 0$

z = 2 * y;

{ $z \geq 0$ } $\{x > 0 \text{ and } z = 2y \text{ and } y = x-1\} \Rightarrow y \geq 0$ and $z = 2y \Rightarrow z \geq 0$

$2(x-1) > 0$

$z = 2y$

z = z + 1;

{ $z \geq 1$ }

$\{x > 0 \text{ and } z = 2y \text{ and } y = x - 1 \text{ and } z = z+1\}$

$\Rightarrow z \geq 0$ +1 $\quad z \geq 1$

b)

{ 2x >= w }

y = w –2;

{ $y \leq x$ }

x = 2*x;

{ $y < x$ && $w \leq x$ && $x \% 2 = 0$ }

z = x-2;

{ $z \% 2 = 0$ && $z \geq y$ }

$\{2x \geq w \text{ and } y = w-2\} \Rightarrow y = 2x - 2$

$y \geq x$

$2x \geq w$

$x \geq w/2$

$2x \geq w-2$ $\quad x = 3 \quad x = 0$

$x \geq w/2 - 2$ $\quad w = 2 \quad w = 0$

$\{x = 2{*}x \text{ and } y = w-2 \text{ and } 2x \geq w\}$

ex! $x=3$, $w=2$ or $x=0$, $w=0$ or $x=1$, $w=2$ $\qquad y < 2x$

$y = 1-2=0 \qquad y = 0-2 \qquad y = 2-2=0$

$y \leq x \qquad y \leq x \qquad y \leq x$

$x=6 \qquad x=0 \qquad x=2$

$y < x \qquad y < x$

$w < x \qquad w \leq x \qquad w \leq x$

$z = 6-2=4 \qquad z = 0-2=-2 \qquad z = 2-2=0$

Name: Aaron Taylor

Question 7. (12 pts) TRUE/FALSE.

a) ([✓ TRUE]/FALSE) If specification A is stronger than specification B, then any implementation that satisfies B satisfies A 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 exactly the same logical formula.)

c) ([✓ TRUE]/FALSE) You can strength a specification by weakening the precondition.

Consider the following code for a binary search:

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

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

d) ([✓ TRUE]/FALSE)

@requires a is sorted and key is contained in a
@modifies nothing   Checks error case ✓
@returns i such that a[i] = key

e) ([✓ TRUE]/FALSE)

@requires a is sorted
@modifies nothing
@returns i such that a[i] = key if such an i exists and a negative value otherwise
@throws IllegalArgumentException   Checks error case

f) (TRUE/[✓ FALSE])

@requires a is not null   a must be sorted
@modifies nothing
@returns returns i such that a[i] = key if such an i exists and a negative value otherwise

Name: Aaron Taylor

Question 8. (12 pts) change these!

Consider the following specifications for a method that has one int argument:

(a) Returns an integer ≥ the argument

(b) Returns a non-negative integer ≥ the argument

(c) Returns argument – 1

(d) Returns argument^2 (i.e., the square of the argument)

(e) Returns a non - negative number

Consider these implementations, where arg is the function argument value:

(i) return arg + 5;

(ii) return arg * arg;

(iii) return arg % 10;

(iv) return Math.abs(arg);

(v) return Integer.MAX_VALUE;

Place a check mark in each box for which the implementation satisfies the specification. If the implementation does not satisfy the specification, leave the box blank

	(a)	(b)	(c)	(d)	(e)
(i)	✓				
(ii)	✓	✓		✓	✓
(iii)					✓
(iv)	✓	✓			✓
(v)	✓	✓			✓

Name: Aaron Taylor

Question 9 (14 pts)

Prove that the given code below computes the correct result if the code terminates. To do so, identify the loop invariant and decrementing function. Next, prove that the loop invariant holds, then prove that the decrementing function is indeed a decrementing function that decreases to zero.

```
int product(int [] arr) {
    n = 1;
    p = arr[0];
    while (n < arr.length) {
        p = p * arr[n];
        n = n + 1;
    }

    Return p;
}
```

rr[n-1]

arr = [1, 2, 3, -3]
n = 3
p = -18

arr = [1, 2, 2, 1, 2]
n = 5
p = 6

P/n > 0

LI: P % (arr[n]) = 0

D: n < arr.length

Base Case: P % (arr[n-1]) = 0

P % (arr[0]) = 0

arr [0] % arr[0] = 0

- Each P_i is a result of P_{i-1} * P[n]
this means that it will divide into the previous array value with no remainder

Assume case holds for n-1

Prove for n:

$P_i = P_{i-1}$ * arr[n_{i-1}]

$n_i = n_{i-1} + 1$

- P_i % (arr[n_i])

(P_{i-1} * arr[n_{i-1}]) % arr[n_{i-1}+1]

(arr[0] * arr[n_{i-1}]) / arr[n_i+1]

results in no remainder

Name: Aaron Taylor

Question 10 (10 points)

Below is the Javadoc specification of the removeRange() method from ArrayList.

protected void removeRange(int fromIndex, int toIndex)
Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive. Shifts any succeeding elements to the left (reduces their index). This call shortens the list by (toIndex - fromIndex) elements. (If toIndex == fromIndex, this operation has no effect.)

Parameters:
fromIndex - index of first element to be removed
toIndex - index after last element to be removed

Throws:
IndexOutOfBoundsException - if fromIndex or toIndex is out of range (fromIndex < 0 || fromIndex >= size() || toIndex > size() || toIndex < fromIndex)

0 1 2
[0, 1, 2, 3, 4, 5, 6]
remov range(3, 5)
4 [0, 1, 2, 5, 6]
6

Rewrite this specification by filling in the clauses below.

Requires: nothing

Modifies: this, size and this[K] forall K:: fromIndex ≤ K < toIndex

Effects: this.size_post = this.size_pre − (toIndex - fromIndex)
forall K:: fromIndex ≤ K < toIndex —> this_post[K] =
= this-pre[K − (toIndex-fromIndex)]
= element

Returns: none

Throws: IndexOut of Bounds exception