Prelim 1 SOLUTION

CS 2110, September 29, 2016, 7:30 PM

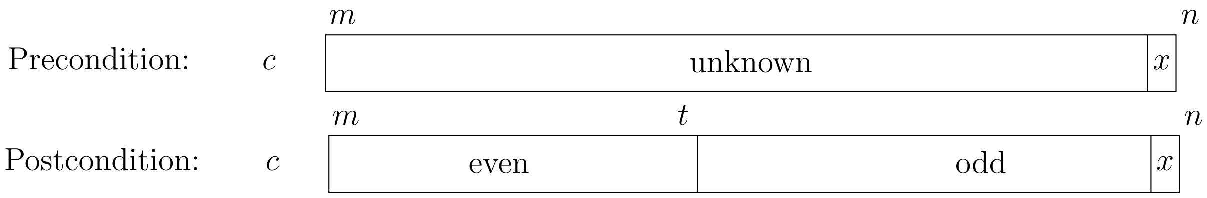
	0	1	2	3	4	5	Total
Question	Name	Loop invariants	Recursion	OO	Short answer	Exception handling	
Max	1	15	15	25	34	10	100
Score							
Grader							

0. Name (1 point)

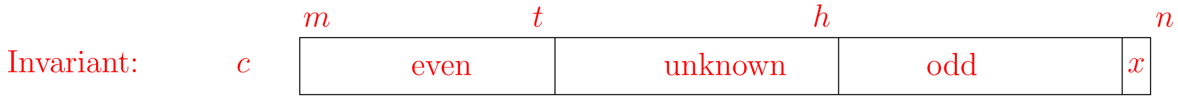
Write your name and NetID at the top of **every** page of this exam.

1. Loop Invariants (15 points)

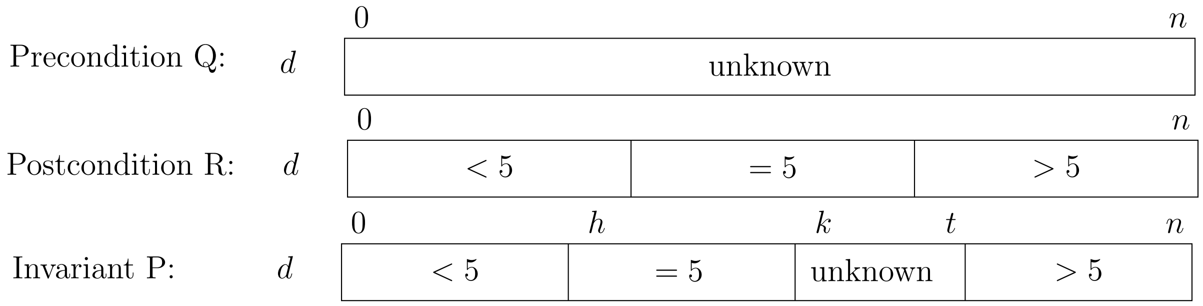
(a) 6 points Consider the following precondition and postcondition.



Generalize them, completing the invariant below. Your generalization should introduce a new variable. Place variables carefully; ambiguous answers will be considered incorrect.



(b) 9 points Consider the following precondition, postcondition, and invariant.



Below, write a loop with initialization that uses invariant P to implement the comment given below. Assume d is already initialized. You don't have to declare variables, but you must assign appropriate values to h, k, and t where necessary. To swap d[i] and d[j], just say, "Swap d[i], d[j]." Your grade depends only on your use of the four loopy questions to write the code.

```
// Given Precondition Q, swap values of d[0..n] to truthify Postcondition R.
h= 0; k= 0; t= n;
while (k <= t) {
    if (d[k] < 5) {Swap d[h] and d[k]; h= h+1; k= k+1;}
    else if d[k] == 5) {k= k+1;}
    else /*dk] > 5*/ {Swap d[k] and d[t]; t= t-1;}
}
```

2. Recursion (15 Points)

(a) Write the following function `occ`, in Java. For example, `occ(2, ’d’)` is ”dd”. You must use recursion. Do not use a loop. Do not write assert statements for the Precondition.

```
/** = a String containing n occurrences of c.
  * Precondition: n >= 0. */
public static String occ(int n, char c) {
    if (n == 0) return "";
    return occ(n-1, c) + c;
}
```

(b) Write the following function `expand`, in Java. Use recursion. Do not use a loop. Do not write assert statements for the Precondition. You can use function `occ`. As an example, p(”2A0B3V”) produces a string with 2 As, 0 Bs, and 3 Vs, i.e. it produces ”AAVVV”.

You will need to convert a character-digit like ’4’ to an int. There are several ways to do that.

```
/** Return s but with each pair "ic" of characters, where i is a digit,
  *  replaced by i occurrences of c.
  *  Precondition: s contains an even number of characters, and the
  *  first of each pair is a digit. */
public static String expand(String s) {
    if (s.length() == 0) return "";
    return occ(s.charAt(0) - ’0’, s.charAt(1))  +  expand(s.substring(2));
}
```

3. Object-Oriented Programming (25 points)

(a) 10 points Below are two class declarations. Complete the bodies of the constructor and function `toString` in class `Grad`. Be careful; pay attention to access modifiers.

```
public class Student {
    private String name;

    /** A student named na.
      * Precon.: no spaces in na. */
    public Student(String na) {
        name= na;
    }

    /** = name of this candidate */
    public String toString() {
        return name;
    }
}
```

```
public class Grad extends Student {
    private String advisor;

    /** Constructor: a grad named
     *  na with advisor ad.
     *  Precon.: no spaces in na, ad. */
    public Grad(String na, String ad) {
        super(na);
        advisor= ad;
    }

    /** = String containing grad's
     * name, a comma and space after
     * it (", "), and advisor. */
    public String toString() {
         return super.toString() + ", " +
                    advisor;
    }
}
```

(b) 5 points Suppose the following assignment has been executed, where the arguments ... are strings containing a person's name and advisor.

```
Grad g= new Grad(..., ...);
```

Write a sequence of statements to extract the advisor of the person in object g and store it in String variable v. You don't have to declare v or any other variables. It doesn't matter whether you wrote methods in part (b) correctly; we go by the method specifications.

```
  String s= g.toString();   // we declared variables
  int t= s.indexOf(" ");    // to help you wth types
  v= s.substring(t+1);      // of variables
}
```

(c) 5 points Method equals, shown below, is to be placed in class Grad. Complete the method body. Also, after the body, write what happens if the type of ob is changed to Grad.

```
/** Return true iff ob is a Grad and
  * ob has the same advisor as this Grad. */
public @Override boolean equals(Object ob) {
    if (!(ob instanceof Grad)) return false;
    return advisor.equals(((Grad)ob).advisor);
}
```

If the type of ob is changed to Grad, the method does not override equals in class Object and the program won't compile because of the @Override annotation.

(d) 5 points Below are two classes and one interface. Below them, state two reasons why this won't compile.

```
public abstract class A implements I {
   public abstract int m();
   public int f= 10;
}

public interface I  {
}
```

```
public class B extends A implements I {
    A a= new A();
    public void p() {
        f= 20;
    }
}
```

(1) Because `A` is abstract, the expression `new A()` won't compile. (2) Because method `m()` in class `A` is abstract, it must be overridden in subclass `B`, but it isn't.

It doesn't matter that interface `I` has no declarations, or that both `A` and `B` implement `I`, or that subclass `B` references public field `f` in superclass `A`.

4. Short Answer (34 points.)

(a) 5 points Write down the steps in executing a method call `m(args)` .

1. Push a frame for the call on the call stack (it contains, among other things, parameters and local variables of `m`).

2. Assign the argument values to the parameters.

3. Execute the method body, using the top frame on the call stack for parameters and local variables.

4. Pop the frame from the stack, and if this is a function push its value onto the call stack.

(b) 5 points. Below are five expressions. To the right of each, write its value.

1. (int)'@' == '@' true. Remember that char is a number type.
2. (char) ('d' - 2) 'b'
3. new Double(5) == new Double(5) false. Each new-expression creates a new object, and the pointers to these objects are different.
4. ((Object)(new Integer(7))).equals(3+4) true
5. (int) 3.5 + 4.1 Reduces to 3 + 4.1, which is 7.1

(c) 5 points. Consider these declarations of classes and interfaces:

```
public class A implements I, J { ... }
public class B extends A implements I { ... }
public interface I { ... }
public interface J { ... }
```

Consider the statement:

```
B var= new B( ... );
```

Write down a list of all things to which variable `var` can be cast. `var` can be cast to Object, A, B, I, and J, in any order and any number of times. For example, this is legal and will work: (A)(I)(I)(J)(B)(B) var

(d) 6 points. Put a check mark before each of the following sentences that is correct and an X before each that is incorrect.

1. In a while loop `while(B) {int x; ...}` , variable x is allocated new space each time the loop body is executed. false. It is allocated space in the first step of executing a method call, pushing a frame on the call stack.

2. In a class `class C {public static int y= 5; ...}` , every time an expression `new C(...)` is evaluated, y is set to 5. false. There is only one copy of a static variable, and it is created and initialized when the program first starts.

3. To make testing easier, Java allows methods in a JUnit testing class to access private fields of objects it is testing. false.

4. If a class implements an interface, its subclasses may also implement that interface. true. There is an example of this in part (c).

5. During execution of a Java program, the call stack contains at most one frame for each method. false. It contains one frame for each call that has started but hasn't completed.

6. If you don't start a constructor body with a call on another constructor, your program will not compile. false. In that case, Java inserts a constructor call for you: super();

(e) 8 points. To the right or below class `C2`, write the output printed by a call on method `main` of class `C2` below. Please be extremely careful.

```
public class C2 {
    private int p= 1;
    private static int q= 2;
    private int m1(int p) { p= q+1; q= q+3; return q; }
    private int m2(int q) { p= q+1; q= q+3; return q; }

    public static void main() {
        C2 c= new C2();
        int x= c.m1(5);
        System.out.println(x + ", " + c.p + ", " + q);
        q= 2; c.p= 1;
        x= c.m2(5);
        System.out.println(x + ", " + c.p + ", " + q);
    }
}
```

5, 1, 5
8, 6, 2

(f) 5 points. Below, write an enum that has the constants AM and PM. Name the enum anything you want.
public enum WhateverAM, PM;

5. Exception handling (10 Points)

Execute the three calls C.me(-1); C.me(0); and C.me(1); on procedure m shown below. You know that calls on System.out.print print on the Console. As you execute the calls on me, place the output of the calls on System.out.print in the places provided on the right below; don't be concerned about starting each print output on a new line.

```
import java.io.*;

public class C {
  public static void me(int p) {
     System.out.print("8. ");
     int y= p / (p - 1);
     try {
        System.out.print("7. ");
        if (p != -1) throw new RuntimeException();
        System.out.print("6. ");
        y= p / 0;
        System.out.print("5. ");
     } catch (ArithmeticException e) {
        System.out.print("4. ");
        if (p == p) throw new RuntimeException();
        System.out.print("3. ");
     } catch (RuntimeException e) {
        System.out.print("2: ");
     }
     System.out.print("1: ");
  }
}
```

CONSOLE FOR C.m(-1): 8. 7. 6. 4.

CONSOLE FOR C.m(0): 8. 7. 2: 1:

CONSOLE FOR C.m(1): 8.