Prelim 1

CS 2110, October 1, 2015, 7:30 PM

	0	1	2	3	4	5	Total
Question	Name	True False	Short Answer	Testing	Strings	Recursion	
Max	1	20	36	16	15	12	100
Score							
Grader							

The exam is closed book and closed notes. Do not begin until instructed.

You have **90 minutes**. Good luck!

Write your name and Cornell **NetID** at the top of **every** page! There are 5 questions on 9 numbered pages, front and back. Check that you have all the pages. When you hand in your exam, make sure your booklet is still stapled together. If not, please use our stapler to reattach all your pages!

We have scrap paper available. If you do a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam, so that we can make sense of what you handed in.

Write your answers in the space provided. Ambiguous answers will be considered incorrect. You should be able to fit your answers easily into the space provided.

In some places, we have abbreviated or condensed code to reduce the number of pages that must be printed for the exam. In others, code has been obfuscated to make the problem more difficult. This does not mean that its good style.

0. Name 1 point

Ensure that your name and NetID is written on **every** page of this exam.

1. true / false (20 points)

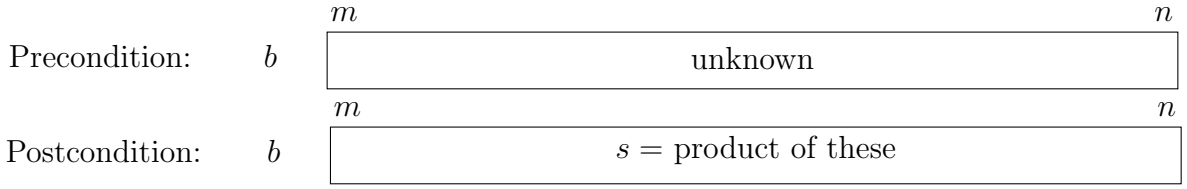
a)	T	F	Evaluation of `"" + a` throws an exception if `a` is `null`.
b)	T	F	`Object` is the only class that does not extend **exactly** one other Java class.
c)	T	F	`int[]` is a primitive type.
d)	T	F	If class `A` defines a method with signature `public boolean equals(A)`, we say that class `A` has **overloaded** method `equals`.
e)	T	F	If class `A` defines a method with signature `public boolean equals(Object)`, we say that class `A` has **overridden** method `equals`.
f)	T	F	Space for a local variable declared in an `if`-statement is allocated only if the `if`-condition is true.
g)	T	F	If class `C` extends abstract class `A`, then the statement `A ob= new C();` is legal.
h)	T	F	Let `s` be a `String` variable. Suppose `s` is `null` and assertions are enabled. Then, execution of `assert s.length() > 0;` results in an assertion error (as opposed to some other error).
i)	T	F	Execution of `C ob= new C(5);` stores the whole new object in `ob`.
j)	T	F	The statement `Set<Integer> b= new Set<Integer>();` is legal because interfaces can be instantiated.
k)	T	F	If class `C` declares a non-static method `m()` and a static field `name`, then `m()` cannot access `name`.
l)	T	F	If a local variable `f` in method `m()` has the same name as a field, and there is no setter method for field `f`, the field cannot be accessed within `m()`.
m)	T	F	Execution of `Point[] t= new Point[4];` stores in `t` a pointer to an array that contains pointers to four newly created `Point` objects.
n)	T	F	If abstract class `A` contains abstract method `m()`, and abstract class `B` extends `A`, then `B` doesn't have to declare `m()`.
o)	T	F	A doubly linked list allows us to access any value in the list in time linear in the length of the list.
p)	T	F	The only reason to make a class abstract is so that you can put abstract methods in it.
q)	T	F	If classes `B` and `D` both extend `A`, the statement `D c= (D) new B();` is illegal.
r)	T	F	Within a constructor, one can use `super(...); this(...);` (with appropriate arguments) to first fill in the superclass fields and second use another constructor in this class to fill in this class's fields.
s)	T	F	Suppose `v` points at an object. Execution of `v= null;` deletes the object to which `v` points.
t)	T	F	If class `Square` extends class `Shape` and `sh` is declared to be of type `Shape`, then `Square sq= (Square)sh;` will compile but might result in an exception at runtime.

2. Short Answer (36 points)

(a) (4 points) Consider the following code segment, note that `(int)'a'` is 97, `(int)'b'` is 98, and `(char)99` is `'c'`. Execute the segment, writing each value printed to the right of the corresponding `println` statement.

```
String a= "a";
char b= 'b';
int c= 99;
System.out.println(a + b + c);
System.out.println(a + b + (char) c);
System.out.println(c + b + a);
System.out.println(b + c + a);
```

(b) (8 points) A program segment that is supposed to store into s the product of values in $b[m..n]$ has the following precondition and postcondition:



A loop with initialization will be written for this problem, using this loop invariant: s is the product of $b[i+1..n]$ —as shown as the the diagram below.

Answer the following four questions, which gives you all the parts of the loop. Be extremely careful! Do not simply write the loop you are thinking of; use the invariant shown above.

(i) What initial assignment to s and i makes the invariant true?

(ii) What condition along with the invariant implies the postcondition?

(iii) In writing the body of the loop, how do you make progress toward termination?

(iv) In writing the body of the loop, how do you keep the invariant true while making progress toward termination?

(c) (6 points) Indicate which of the following three statements about linked lists are true. Assume you have direct access only to the head of a singly linked list or the head and tail of a doubly linked list. Circle **all** that are true.

(i) Finding the smallest element in a sorted singly linked list takes constant time.

(ii) Inserting a value in the middle of a doubly linked list takes constant time.

(iii) Inserting a value after the third element of a doubly linked list takes time proportional to the length of the list.

(d) (6 points) Consider class A, below. Write down the statements S1, S2, S3, S4, S5, and S6 that are executed, in the order they are executed, for the following procedure calls:

1. first(7); Statements executed:

2. first(0); Statements executed:

3. first(3); Statements executed:

```
public class A {
    public static void first(int i) {
            second(i);
            S6;
    }

    public static void second(int i) {
        S5;
        try {
            int b= 5/i;
            S4;
            if (i == 7) throw new Exception();
            S3;
        } catch (ArithmeticException e) {
            S2;
        }
        S1
    }
}
```

(e) (4 points) Write down the steps in executing a procedure call.

(f) (4 points) Consider the following classes.

```
class D {
    public String toString() {
        return "I'm a D";
    }
}
class B extends D {
    public String toString() {
        return "I'm a B";
    }
    public static void main(String[] args) {
        B b= new B();     // Suppose the value stored in b is B@20
        Object ob= b;
        D d= b;
        System.out.println(b.toString());
        System.out.println(ob.toString());
        System.out.println(d.toString());
    }
}
```

What is printed to the console when running class `B` as an application?

(g) (4 points) Consider the following class.

```
public class {
    public static final Suit CLUBS= new Suit();
    public static final Suit SPADES= new Suit();
    public static final Suit HEARTS= new Suit();
    public static final Suit DIAMONDS= new Suit();
    private Suit() {}
}
```

Is there a better way to represent Suits in Java? If so, rewrite class `Suit` below in that way. If not, explain briefly why the above class is the best way to represent Suits.

3. Testing (16 points)

(a) 8 points Consider the following code, which computes the median of an array of `int`s. Recall that the median of a collection is the "middle" value if the collection has an odd number of elements or the arithmetic mean of two "middle" values otherwise.

```
/** Return the median value in b.
  * Precondition: b is not null and b.length >= 1.  */
public static double median(int[] b) {
    int n= b.length;
    if (n % 2 != 0)  // b has an odd number of elements
        return b[n/2];
    // b has an even number of elements
    return (b[n/2-1] + b[n/2]) / 2.0;
}
```

- Unfortunately this code has a serious bug. Write a test to illustrate the bug:

```
@Test
public void testMedian() {









}
```

- Modify the precondition so the method meets its specification.

(b) 8 points The following method computes c raised to the power n.

```
/** Return c^n.
* Precondition: n >= 0.  */
public static double power(double c, int n) {
    if (n == 0) return 1;
    // n > 0
    return power(c * c, n / 2);
}
```

- Unfortunately this code has a serious bug. Write a test to illustrate the bug:

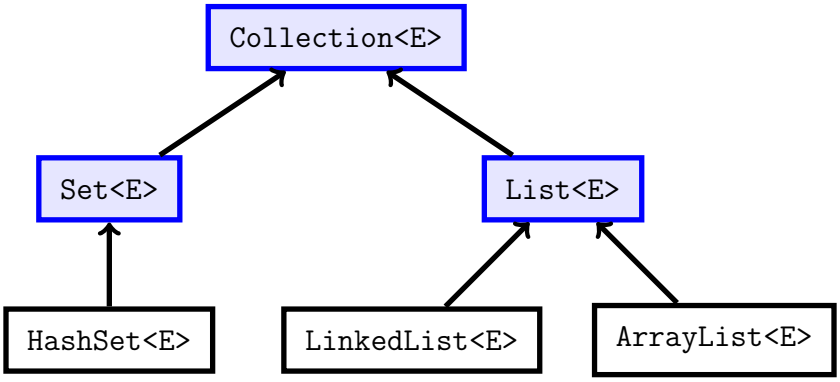
```
@Test
public void testPower() {









}
```

- Modify the **code** so the method meets its specification:

```
/** Return c^n.
  * Precondition: n >= 0.    */
public static double power(double c, int n) {









}
```

Reference: Java Collections

Following is a reference to selected classes (unshaded) and interfaces (shaded) from the `Collections` library, as well as the most important methods defined in the interfaces. This information may be useful in completing the exercise on the opposite page.




```
interface Collection<E> {
  boolean add(E);
  void clear();
  boolean contains(Object);
  boolean equals(Object);
  boolean isEmpty();
  boolean remove(Object);
  int size();
  ...
}

interface Set<E> extends Collection<E> {
  ...
}

interface List<E> extends Collection<E> {
  boolean add(int, E);
  E get(int);
  E remove(int);
  E set(int, E);
  ...
}
```

To iterate over a collection, use a `for` loop:

```
Collection<E> c = ...;
for(E e : c) {
   ...
}
```

4. Strings and Collections (15 points)

(a) 7 points Suppose we have a method `areAnagrams` (you wrote such a method in A2):

```
/** Return true iff s and t are anagrams of each other. */
public static boolean areAnagrams(String s, String t) { ... }
```

Complete method `anagramSet` below according to its specification. Note that `Set` is an interface, so you will have to use an implementing class to construct the actual set.

```
/** Return a set of Strings in l that are anagrams of s.
  * Precondition: l != null, s != null.  */
public static Set<String> anagramSet(ArrayList<String> l, String s) {









}
```

(b) 8 points Complete method `getChars` below according to its specification.

```
/** Return a list of the characters is, in the same order, but with
  * blanks ' ' removed. Example:
  * getChars(['a', ' ', ' ', 'b', 'c', ' ', 'c' ]) is ['a', 'b', 'c', 'c'] */
public static ArrayList<Character> getChars(char[] b) {









}
```

5. Recursion (12 Points)

(a) 6 points Define define the fosterial function $f(n, k)$ as follows:

- $f(1, k) = f(0, k) = 1$
- If $n > 1$ and n is not divisible by k, $f(n, k)$ is the product of $f(n-1, k)$ and $f(n-2, k)$.
- If $n > 1$ and n is divisible by k, $f(n, k)$ is the product of $f(n-1, k)$, and n.

Complete function `fosterial` below according to its specification.

```
/** Return the fosterial value f(n,k).
  * Precondition: n >= 0, k > 0 */
public static int fosterial(int n, int k) {




}
```

(b) 6 points Complete the following function body. Do not change `n` into a `String` or other `char`-valued thing. Work only with type `int`.

```
/** Return true iff the decimal representation of n has only 5's in it.
  * Example hasOnly5(515) is false, hasOnly5(5555) is true.
  * Precondition: n >= 0. */
public static boolean hasOnly5(int n) {




}
```