Exam 1 (Practice)
(February 24, 2017)

This exam is closed-book, notes, and technology.
Please do not open the test until the instructor says time has begun.
Please stop writing once the instructor has called time.
Failure to stop writing will result in a zero on the exam.

Remember you are here to learn.
Relax and think of this as yet another learning experience.

Good luck, have fun!

UID (Not Your Name):

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/ 15</td>
</tr>
<tr>
<td>2</td>
<td>/ 15</td>
</tr>
<tr>
<td>3</td>
<td>/ 15</td>
</tr>
<tr>
<td>4</td>
<td>/ 15</td>
</tr>
<tr>
<td>5</td>
<td>/ 20</td>
</tr>
<tr>
<td>6</td>
<td>/ 20</td>
</tr>
<tr>
<td>Total</td>
<td>/ 100</td>
</tr>
</tbody>
</table>
Problem 1: Trace Amounts of Fun

Consider the following Java code:

```java
public class C {
    public int x;
    public D d;
    public C(int x, D d) {
        this.x = x;
        this.d = d;
    }
}

public class D {
    public int[] arr;
    // Hint: what do the elements of
    // the array default-init to?
    public D() { arr = new int[3]; }
}

public static void fun1(int x) {
    D d = new D();
    C c = new C(x, d);
    // Point A
    fun2(c, 10);
    // Point D
}

public static void fun2(C c, int x) {
    // Point B
    c.d = new D();
    for (int i = 0; i < 3; i++) {
        c.d.arr[i] = x + c.c;
    }
    // Point C
}
```

Give complete stack-and-heap diagrams outlining the state of memory at each of the given points above. Assume that program execution begins with a call to `fun1(-1)`. The style of your diagrams should be identical to that presented in class. You should reproduce the diagrams completely at each point of the program. (Hint: remember that objects on the heap possess a tag indicating their type.)
Problem 2: I Have a Proposition for You

Consider the following Java method:

```java
// pre: arr1 != null, arr2 != null, arr1 and arr2 are sorted
public static int[] calculate(int[] arr1, int[] arr2) {
    int[] arr3 = new int[arr1.length + arr2.length];
    int i = 0;
    int j = 0;
    int k = 0;
    // Point A
    while (i < arr1.length && j < arr2.length) {
        if (arr1[i] < arr2[j]) {
            arr3[k++] = arr1[i++];
        } else {
            arr3[k++] = arr2[j++];
        // Point B
        }
    }
    while (i < arr1.length) { arr3[k++] = arr1[i++]; }
    while (j < arr2.length) { arr3[k++] = arr2[j++]; }
    // Point C
    return arr3;
}
```

For each of the propositions below, determine if the proposition never holds (×), sometimes holds (?), or always holds (✓) at the given program points. (Hint: The third proposition considers all valid indices of arr1, i.e., 0, ..., arr1.length - 1.)

<table>
<thead>
<tr>
<th>Proposition</th>
<th>Condition 1</th>
<th>Condition 2</th>
<th>Condition 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1 &lt; arr1.length</td>
<td>k &lt;= arr2.length</td>
<td>forall i. arr1[i] &lt;= arr3[arr3.length - 1]</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In a sentence or two, describe what the method returns as output:
Problem 3: Quick and Dirty

(a) Consider the following Java interface and classes:

```java
public interface I { public void f(); }
public class C1 implements I {
    public void f() { System.out.println("C1.f"); }
}
public class C2 implements I {
    public void f() { System.out.println("C2.f"); }
    public void g() { System.out.println("C2.g"); }
}
```

For each of the following variable declarations and method calls, determine if the method call typechecks. If it does, say what the method call prints to the console; if it does not, say in one sentence why it does not typecheck.

- `I i = new C1(); i.f();` ____________________________

- `I i = new C2(); i.g();` ____________________________

(b) Consider the following Java class that implements a one-element cell:

```java
public class Box {
    private int value;

    public Box(int value) { this.value = value; }

    public int getValue() { return value; }

    public void setValue(int value) { this.value = value; }
}
```

Modify the `Box` class so that it is generic in the type of the element that the box contains. Inline your changes into the code listing above.

```java
public class Box<T> {
    private T value;

    public Box(T value) { this.value = value; }

    public T getValue() { return value; }

    public void setValue(T value) { this.value = value; }
}
```
Problem 4: Developing a Complex

For each of the following methods:

1. Give a mathematical function or recurrence that models the time complexity of the method. State explicitly what operations your function tracks as well as what the input to the function represents. If your model is a recurrence relation, solve that relation for an explicit mathematical function.

2. Give a tight upper-bound for your function using Big-O notation. You can simply state the upper-bound rather than formally proving it correct.

```java
public static int f1(int n, int k) {
    int[] ret = new int[n];
    for (int i = 0; i < n; i++) {
        ret[i] = k;
    }
    return ret;
}
```

```java
public static String f2(int n) {
    if (n == 0) {
        return "";
    } else {
        return f2(n / 10) + (n % 10);
    }
}
```

// Hint: think carefully about what f2 actually computes with respect to its input...
Problem 5: Fundies

Write a complete Java class, Complex, that represents a complex number. Recall that a complex number is made up of two numbers: a real part $a$ and an imaginary part $b$. It is typically written in the form $a + bi$ where $i = \sqrt{-1}$. Your class should support the following constructor and methods:

- **Complex(int a, int b)**: constructs a new complex number of the form $a + bi$.
- **Complex add(Complex other)**: adds this complex number to the other complex number, returning a new complex number that is the result. Complex number addition is defined as follows: $(a_1 + b_1i) + (a_2 + b_2i) = (a_1 + a_2) + (b_1 + b_2)i$.
- **Complex conjugate()**: returns new a complex number that is the conjugate of this one. The conjugate of a complex number $a + bi$ is defined to be $a - bi$.
- **double abs()**: returns the absolute value of this complex number. The absolute value of a complex number $z = a + bi$ is defined to be $|z| = \sqrt{a^2 + b^2}$.
- **String toString()**: returns the string representation of this complex number: "$a + bi$".

To write these functions, your code should import the java.lang.Math class and use the following static functions:

- **double Math.sqrt(double x)**: returns $\sqrt{x}$.
- **double Math.pow(double a, double b)**: return $a^b$. 

Problem 6:  A Link to the Past

Consider the following implementation of a generic linked list:

```java
public class Node<T> {
    public T value;
    public Node<T> next;
    public Node(T value, Node<T> next) {
        this.value = value;
        this.next = next;
    }
}
```

```java
public class LinkedList<T> {
    private Node<T> first;
    // ...
}
```

Write a method for the LinkedList<T> class called dedupHead() that removes every element of the list that is the same as the first element of the list. The first element of the list always remains in the list. For example, if the list contains the elements [3, 1, 3, 8, 9, 10, 4, 3], then dedupHead() modifies the list so that it becomes [3, 1, 8, 9, 10, 4]. You may not use any additional methods of the LinkedList<T> class in your code.

In addition to the implementation, state the worst case time complexity of the method using Big-O notation in terms of the length of the list n; you don’t need to formally prove this bound.

(Hint: use the equals method to test to see if elements of your list are equal.)