CS 2110 Pretest

0. Write your name and net ID on each page.

1. (10 Points) Consider the class definitions for Item, Book, and Author shown in box below and the following code, which creates several objects of type Item, Book, and Author.

   ```java
   Author a1 = new Author("Stephen King");
   Author a2 = new Author("Stephen King");
   Item b1 = new Book("It", a1);
   Book b2 = new Book("Carrie", a1);
   Book b3 = (Book) b1;
   Item b4 = new Item("Chotchkie");
   Book b5 = new Book("It", a2);
   b1.placeOrder(10);
   b2.placeOrder(51);
   b5.placeOrder(17);
   
   Evaluate the following expressions and write their values to the right of the expression. If the statements contain an error, describe the error and when it would occur (at the time of compilation or at runtime).

   a. b1.equals(b5)                                    false
   b. b1 == b5                                         false
   c. Item.compare(b1, b5)                            true
   d. b1.equals(b3)                                    true
   e. b1 == b3                                         true
   f. Item.compare(b1, b3)                            true
   g. b2.getAuthorReputation()                        6.0
   h. b3.getAuthorReputation()                        6.0
   i. b5.getAuthorReputation()                        1.0
   j. ((Book)b4).getAuthorReputation()                runtime ClassCastException
   k. b1.getAuthorReputation()                        compile error: no such method
class Item {
    private String name;
    private int totalSales = 0;
    public Item(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public String toString() {
        return "Item: " + name;
    }
    public void placeOrder(int quantity) {
        totalSales += quantity;
    }
    public static boolean compare(Item itemOne, Item itemTwo) {
        return itemOne.toString().equals(itemTwo.toString());
    }
}

class Book extends Item {
    private Author author;
    public Book(String title, Author author) {
        super(title);
        this.author = author;
    }
    public String toString() {
        return "Book: " + getName() + " by " + author.getName();
    }
    public void placeOrder(int quantity) {
        super.placeOrder(quantity);
        author.updateReputation(quantity);
    }
    public double getAuthorReputation() {
        return author.getReputation();
    }
}

class Author {
    private String authorName;
    private int numSold = 0;
    public Author(String name) {
        authorName = name;
    }
    public String getName() {
        return authorName;
    }
    public void updateReputation(int quantity) {
        numSold += quantity;
    }
    public double getReputation() {
        return numSold / 10;
    }
}
2. (10 Points) Review the definitions of class Student and class CornellStudent. It is not necessary to add methods to either class definition, but if you want to use additional methods, you may define them as part of your answer (be sure to specify to which class the method belongs).

a. In the space below, implement the equals method of Student.

```java
return (other instanceof Student &&
        ((Student)other).name.equals(name));
```

b. In the space below, implement the constructor of CornellStudent.

```java
super(studentName);
this.netId = "abc1";
setStudyLocation("CTB");
```
c. In the space below, implement the equals method of CornellStudent.

```java
return other instanceof CornellStudent
    && super.equals(other)
    && this.netId.equals(((CornellStudent)other).netId);
```

3. (10 Points) Use recursion to implement the method hasName, defined below.

```java
class Person {
    private String name;    // name of this person
    private Person mother;  // this person's mother, null if unknown
    private Person father;  // this person's mother, null if unknown

    /** = “this Person or one its ancestors has name n.” */
    private boolean hasName(String n) {
        if (name.equals(n)) {
            return true;
        }
        if (mother != null && mother.hasName(n)) {
            return true;
        }
        if (father != null && father.hasName(n)) {
            return true;
        }
        return false;
    }
}
```
4. (10 Points) Arrays. Implement the method merge, defined below.

/**
 * Merge takes two arrays, assumed to be in sorted order, and
 * merges them into a single array, also in sorted order.
 * If the two arrays have length x and y, then the merged array
 * has length x + y.
 *
 * Example: on input arrays [10, 11, 14] and [12, 13, 15] merge
 * would return the array [10, 11, 12, 13, 14, 15].
 */
public static int[] merge(int[] firstArr, int[] secondArr) {

    int[] mergedArr = new int[firstArr.length + secondArr.length];
    int idx1 = 0;
    int idx2 = 0;
    for (int mIdx = 0; mIdx < mergedArr.length; mIdx++) {
        if (idx2 == secondArr.length ||
            (idx1 < firstArr.length &&
             firstArr[idx1] < secondArr[idx2])) {
            mergedArr[mIdx] = firstArr[idx1];
            idx1 += 1;
        } else {
            mergedArr[mIdx] = secondArr[idx2];
            idx2 += 1;
        }
    }
    return mergedArr;
}