Part A: 16 points

What is printed by each `System.out.printf` statement? There are four such statements, each prints four values. Write the output for each `printf` statement immediately below it.

```java
String s = new String("blue");
String t = s;
String u = "duke" + t;
System.out.printf("%s %s %s %d\n", s, t, u, s.length());

s = "red";
ut = t + t;
System.out.printf("%s %s %s %d\n", s, t, u, u.length());

String[] a = {"cat", "bird", "dog", "tiger");
String[] b = a;
a[2] = "fish";
b[0] = "bat";
System.out.printf("%s %s %s %d\n", a[0], a[1], b[2], b[3].length());

HashMap<String, Integer> map = new HashMap<>();
map.put("to", 2);
map.put("in", 7);
map.put("me", 4);
map.put("to", 5);
map.put("in", map.get("in")+2);
System.out.printf("%d %d %d\n", map.get("to"), map.get("in"), map.get("me"), map.size());
```
Part B: 8 points
Write one of: True, False, or cannot be determined. Explain/Justify each answer

1. If two string variables s and t have s.hashCode() == t.hashCode(), that is, they have the same hash code values, then what is the value of s.equals(t)?

2. Suppose there are two string variables s and t such that s.equals(t) is false. What is the value of s.hashCode() == t.hashCode()?

3. Suppose there are two string variables s, with value "hello", and t with value "hello". What is the value of s == t?

4. Suppose there are two string variables s, with value "hello", and t with value "hello". What is the value of s.hashCode() == t.hashCode()?
PROBLEM 2:  \textit{(NBody goes round and round (8 points))}

The three sections of code below are from a version of the \texttt{CelestialBody} class. The top left is the start of the class showing some of its state, the bottom left is the method \texttt{getX}, and on the right is the method \texttt{calcDistance}.

\textbf{Part A.} What type of value is returned by the method \texttt{calcDistance}?

\textbf{Part B.} Can line 107 in \texttt{calcDistance} be replaced by the following line? Explain.
\begin{verbatim}
double dx = this.myXPos - b.myXPos;
\end{verbatim}

\textbf{Part C.} Can line 107 in \texttt{calcDistance} be replaced by the following line? Explain.
\begin{verbatim}
double dx = myXPos - this.b.myXPos;
\end{verbatim}

\textbf{Part D.} Can line 107 in \texttt{calcDistance} be replaced by the following line? Explain.
\begin{verbatim}
double dx = getX() - b.myXPos;
\end{verbatim}
As an example of how to think about some of the questions in this section, consider the method `stuff` below. The runtime complexity of this method is $O(n)$ and the value returned by the function is $O(n^2)$ for parameter $n$. As a concrete example, note than when $n = 100$ the loop executes 100 times doing an $O(1)$ operation each time. The value returned is $1 + 2 + \ldots + 99 = (98 \times 99)/2$. Note that even if the return statement was `return sum*2` that the value returned would still be $O(n^2)$.

```java
public int stuff(int n){
    int sum = 0;
    for(int k=0; k < n; k++){
        sum += k;
    }
    return sum;
}
```

In all these problems $n$ is a positive number. In each problem you should provide two big-Oh expressions: one for runtime and one for the value returned. Briefly justify each answer you provide. Your answers are for the entire method, justification can include discussing lines/loops in each method.

**Part A (4 points)**

```java
public int misc(int n){
    int sum = 0;
    for(int k=0; k < n; k++){
        sum += k;
        sum += n;
    }
    return sum;
}
```

a) What is the runtime complexity of method `misc` in terms of $n$? Use big-Oh and justify your answer briefly.

b) What is the value returned by method `misc` in terms of $n$? Use big-Oh and justify your answer briefly.
Part B (4 points)
If it helps, for this problem you can assume $n$ is a power of 2.

```java
public int something(int n)
{
    int sum = 0;
    for(int k=1; k <= n; k *= 2)
    {
        for(int j=1; j <= n; j *= 2) {
            sum += 1;
        }
    }
    return sum;
}
```

a) What is the runtime complexity of method `something` in terms of $n$? Use big-Oh and justify your answer briefly.

b) What is the value returned by method `something` in terms of $n$? Use big-Oh and justify your answer briefly.
Part C (4 points)

public int eval(int n){
    int sum = 0;
    for(int k=0; k < n; k += 1){
        for(int j=k; j < n; j += 1) {
            sum += n;
        }
    }
    for(int k=0; k < 2*n; k += 1){
        sum += n;
    }
    return sum;
}

a) What is the runtime complexity of method eval in terms of $n$? Use big-Oh and justify your answer briefly.

b) What is the value returned by method eval in terms of $n$? Use big-Oh and justify your answer briefly.
PROBLEM 4:  (Storing data in two ways (36 points))

Consider the class named Storage that is shown on the next two pages. This class sets aside a fixed amount of storage and can store words associated with a variable of type Storage in two ways, in an array with duplicates and in a set.

The class Storage has four private variables:

```java
private String [] myItems;
private HashSet<String> myUniqueItems;
private int mySize; // total current number in myItems
private int myCapacity; // total capacity in myItems
```

Consider the following example showing numbered lines.

9:  String [] itemsA = {"flour", "sugar", "butter", "sugar", "salt", "butter"};
10: Storage a = new Storage(10, itemsA);

When line 10 is executed, Storage a’s private variable myItems will result in having the following values (with index values shown). Note that no values were put in index positions 6 through 9.

<table>
<thead>
<tr>
<th>&quot;flour&quot;</th>
<th>&quot;sugar&quot;</th>
<th>&quot;butter&quot;</th>
<th>&quot;sugar&quot;</th>
<th>&quot;salt&quot;</th>
<th>&quot;butter&quot;</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Storage a’s myUniqueItems would contain four unique items: "flour", "sugar", "butter", "salt".

Storage a’s mySize would have value 6, as 6 strings were put in, and Storage a’s myCapacity would have value 10.

Complete this problem on the remaining pages.

This problem has 9 parts labeled A through I.
import java.util.*;

public class Storage {
    private String[] myItems;
    private HashSet<String> myUniqueItems;
    private int mySize; // total current number in myItems
    private int myCapacity; // total capacity in myItems

    public Storage() {
        mySize = 0;
        myCapacity = 1000;
        myItems = new String[1000];
        myUniqueItems = new HashSet<String>();
    }

    public Storage(int capacity, String[] items) {
        myCapacity = capacity;
        mySize = items.length;
        if (mySize > capacity) {
            myCapacity = mySize;
        }
        // rest of code not shown
    }

    public int getMyCapacity() {
        // code not shown
    }

    public int getMySize() {
        // code not shown
    }

    public void add(String item) {
        if (mySize >= myCapacity) {
            throw new IndexOutOfBoundsException("bad index in add "+mySize);
        }
        // code not shown
    }

    public boolean updateCapacity(int newCapacity) {
        if (newCapacity <= myCapacity) {
            return false;
        }
        // code not shown
        return true;
    }

    @Override
    public String toString() {
        // change code
        return String.format("(size: %d, capacity: %d)", mySize, myCapacity);
    }
}
```
public int howMany(String word) {
    return 1;
}

@Override
public boolean equals(Object o) {
    Storage other = (Storage) o;
    // code not shown
    return false;
}

@Override
public int hashCode() {
    return this.toString().hashCode();
}
```
Part A. (4 points)

Complete the Storage constructor shown in lines 18-25 in the full class and started below. This constructor receives an integer named capacity and a String array named items. The private variable mySize has been set to the number of items in items and the private variable myCapacity has been set to the value capacity passed in and adjusted if needed. Each element from items should be stored in both myItems and myUniqueItems. In myItems they should be stored in the same order they are in items. Complete the code to initialize myItems and myUniqueItems.

```java
public Storage(int capacity, String[] items) {
    myCapacity = capacity;
    mySize = items.length;
    if (mySize > capacity) {
        myCapacity = mySize;
    }
}
```

Part B. (2 points) Complete the Storage method named getMyCapacity shown on lines 28-30 that has no parameters. This method should return the capacity of the Storage. In the example at the beginning of this problem, if the next line were to print the value of a.getMyCapacity(), then 10 would be printed.

```java
public int getMyCapacity() {
}
```
Part C. (2 points) Complete the Storage method named `getMySize` shown on lines 33-35 that has no parameters. This method should return the number of elements stored in `myItems` (don’t include slots that are not storing values). In the example at the beginning of this problem, if the next line were to print the value of `a.getMySize()`, then 6 would be printed.

```java
public int getMySize() {
}
```

Part D. (6 points)
Currently the `toString` method shown on lines 56-60 prints out and identifies the values of `mySize` and `myCapacity`. Instead it should print out and identify the values of `mySize`, the size of `myUniqueItems`, `myCapacity` and the unique strings in the Storage in sorted order. For example, for the lines

9: String [] itemsA = {"flour", "sugar", "butter", "sugar", "salt", "butter"};
10: Storage a = new Storage(10, itemsA);
11: System.out.printf("%s\n", a);

the `toString` method should print in this format (note items are sorted):

(size: 6, unique: 4, capacity: 10, items: butter flour salt sugar)

Complete the `toString` method below. The old return has been commented out.

```java
@Override
public String toString() {
// return String.format("(size: %d, capacity: %d)", mySize, myCapacity);
}
Complete the Storage method named add shown on lines 38-43 that has one parameter, a String named item. This method should add item to the Storage. For example

9: String [] itemsA = {"flour", "sugar", "butter", "sugar", "salt", "butter"};
10: Storage a = new Storage(10, itemsA);
11: a.add("sugar");
12: a.add("egg");

After line 12 is executed, Storage a’s private variable myItems will result in having the following values (with index values shown). Note that "sugar" and "egg" were added to slots 6 and 7, and no values were put in index positions 8 through 9.

<table>
<thead>
<tr>
<th>&quot;flour&quot;</th>
<th>&quot;sugar&quot;</th>
<th>&quot;butter&quot;</th>
<th>&quot;sugar&quot;</th>
<th>&quot;salt&quot;</th>
<th>&quot;butter&quot;</th>
<th>&quot;sugar&quot;</th>
<th>&quot;egg&quot;</th>
<th>—</th>
<th>—</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Storage a’s myUniqueItems would contain five unique items: "flour", "sugar", "butter", "salt", "egg".

Storage a’s mySize would have value 8, as 8 strings were put in, and Storage a’s myCapacity would have value 10.

Complete the method below.

```java
public void add(String item) {
    if (mySize >= myCapacity) {
        throw new IndexOutOfBoundsException("bad index in add "+mySize);
    }
}
```
Part F. (6 pts)

Complete the Storage method named updateCapacity shown on lines 46-53 that has one parameter, an int named newCapacity. This method should change the capacity of myItems to the newCapacity indicated if newCapacity is larger than the current capacity. For example in the code below, Storage a’s capacity is changed to 20. This method returns true if it updated the capacity, and returns false if it did not update the capacity.

```java
9: String [] itemsA = {"flour", "sugar", "butter", "sugar", "salt", "butter"};
10: Storage a = new Storage(10, itemsA);
11: if (! a.updateCapacity(20))
12: System.out.printf("Value to low to update capacity\n");
```

Complete the method below.

```java
public boolean updateCapacity(int newCapacity) {
    if (newCapacity <= myCapacity ) {
        return false;
    }
    return true;
}
```

return true;
Part G. (4 pts)
Complete the Storage method named howMany shown on lines 65-67 that has one parameter, a String named word. This method should return the number of times word appears in myItems. Currently it does not work correctly, it always returns 1. In the example below, if implemented correctly, line 11 prints 2, as "butter" appears twice in Storage a.

9: String [] itemsA = {"flour", "sugar", "butter", "sugar", "salt", "butter"};
10: Storage a = new Storage(10, itemsA);
11: System.out.printf("%d\n", a.howMany("butter"));

Complete the method below.

    public int howMany(String word) {
Part H. (4 pts)
Complete the Storage method named equals shown on lines 70-76 that has one parameter, an Object named o. This method should correctly compare two Storage objects. Two Storage objects are equal if their capacities are equal, if the number of items in mySize is equal and if the unique items are the same. In the example below, a.equals(b) is true, they have the same mySize, even though they have different numbers of each items, and b.equals(c) is false, they have the same number of each items but the capacities are different.

```
9: String [] itemsA = {"flour", "sugar", "butter", "sugar", "salt", "butter"};
10: Storage a = new Storage(10, itemsA);
11: String [] itemsB = {"sugar", "butter", "salt", "flour", "salt", "salt"};
12: Storage b = new Storage(10, itemsB);
13: Storage c = new Storage(20, itemsB);
```

Complete the method below.

```java
@Override
public boolean equals( Object o) {
    Storage other = (Storage) o;
    // Add your implementation here
}
```

Part I. (3 pts)
Consider the Storage method named hashcode shown on lines 79-82 and also shown below.

```java
@Override
public int hashCode() {
    return this.toString().hashCode();
}
```

Compare the value returned to replacing it with the line return 10;. Compare the two and explain which hashcode is better.