Question 1. [7 MARKS]

Suppose you have a repository containing one file called `birthdays.txt`, with the following contents:

```
Jane - July 23
Mia - July 15
Rohit - Feb 10
```

and that two partners have checked it out in two different places: Partner 1 has working copy 1 and partner 2 has working copy 2 (or WC1 and WC2 for short). In the following questions, assume that the *only* subversion commands executed are the ones described.

Part (a) [1 MARK]

Suppose that, inside working copy 1, partner 1 changes the line containing Mia's birthday to say `July 17` and then does `svn commit -m "Corrected Mia's birthday"`. Inside WC2, what would partner 2 see when executing `svn status`?

```
Nothing at all.
Although the change in WC1 was committed to the repository, svn status executed in WC2
only reports on things in WC2 that the repository doesn't know about.  It doesn't report
on things in the repository that WC2 doesn't know about.
```

Part (b) [6 MARKS]

Suppose next that, inside working copy 2, partner 2 changes the line containing Mia's birthday to say `July 30` then does `svn commit -m "Fixed Mia's birthday"`. The commit fails, saying "File or directory 'birthdays.txt' is out of date; try updating". But when partner 2 tries `svn update`, svn reports that there is a conflict.

Suppose Partner 2 is sure about the birthday. Show the steps that must be taken in order for working copy 1 to also have Mia's birthday as July 30. For each step, show where it must be done (either in WC1 or WC2).

Where (either WC1 or WC2)	svn command or other action
In WC2	edit birthdays.txt to replace everything between the chevrons (and the chevrons too) with the line containing July 30
In WC2	svn resolved birthdays.txt
In WC2	svn commit "Fixed Mia's birthday" (or whatever log message)
In WC1	svn update

Question 2. [4 MARKS]

Consider the following code:

```
1  public class Miscellaneous {

2     public static void main(String[] args) {

3        Object[] stuff;

4

5        stuff[0] =

6        stuff[1] =

7        stuff[1].append("!!");

8     }

9 }
```

Part (a) [1 MARK]

There is no way to complete line 5 so that the code will compile, unless a certain action is taken before line 5. On line 4 above, write a line of Java code that will solve this problem.

```
4       stuff = new Object[2];      // Or it could have a larger capacity.
```

Part (b) [1 MARK]

Complete line 5 so that it puts the first digit of your phone number into the array.

```
5        stuff[0] = new Integer(6);
```

Part (c) [1 MARK]

Complete line 6 so that it puts a sentence of your choosing into the array. Use a `StringBuffer`, because we are going to append to that sentence on the next line.

```
6        stuff[1] = new StringBuffer("Happy Thanksgiving!");
```

Part (d) [1 MARK]

Line 7 will not compile. Why not?

```
The compiler only knows that Stuff[1] is an Object, and there is no append method in the
Object class.  (We can fix this by casting to String.)
```

Question 3. [5 marks]

Consider the following code from Assignment 0:

```
public class Node {

    // The left and right children of this Node.
    Node left, right;
    // The contents of this Node, if it is a leaf node, or null otherwise.
    Object contents;
    // The parent of this Node, or null if it is the root.
    Node parent;

    // Constructors omitted.
}

public class Encoder {

    /*
     * The prefix code is represented by a binary tree in which each left branch
     * represents a 0, each right branch represents a 1, and the sequence of 0s
     * and 1s represented by the path from root to a leaf is the encoding for
     * the character stored in the leaf.
     */

    // The root of the tree representing this prefix code.
    private Node root = null;
    // A mapping from characters to their corresponding leaf node in the tree.
    private HashMap<Character, Node> lookup = new HashMap<Character, Node>();

    // Methods omitted.

}
```

Part (a) [1 mark]

Does the following describe a legal prefix-free code, as defined on assignment 0?

character	encoding
K	01
L	11
M	10
N	0

Circle one: Yes [✓ No]

```
No character's encoding should be a prefix of any other character's encoding.
In this scheme, the encoding for N is a prefix of the encoding for K.
```

Part (b) [4 marks]

Below, write a public method named `endsInOne()` to go in this class. It should take a `Character` as an argument and return true iff the code for that character ends with a 1.
Hint: This does not require a loop or recursion or anywhere near this full page!

```
public boolean endsInOne(Character c) {

    Node leaf = lookup.get(c);

    return leaf == leaf.parent.right;

}
```

Last Name: ____________________ **First Name:** ____________________

Some parts of the Java API:

```
Class StringBuffer: A mutable sequence of characters.

StringBuffer(String str)
          Constructs a string buffer initialized to the contents of the specified string.
StringBuffer append(String str)
          Appends the specified string to this character sequence.
StringBuffer append(StringBuffer sb)
          Appends the specified StringBuffer to this sequence.


Class HashMap<K,V>: A mapping of keys to values.

HashMap()
          Constructs an empty HashMap.
V put(K key, V value)
          Associates the specified value with the specified key in this map.
V get(Object key)
          Returns the value to which the specified key is mapped, or null if this map
          contains no mapping for the key.
```