Your five-digit number: _____

Problem 0 (1 point)

Put your secret number on each page. Also make sure you have provided the information requested on the first page.

Problem 1 (3 points)

Draw the box-and-arrow diagram that results from executing the main method, including all boxes created anywhere in the code.

```
private static times2 (Point p) {
    p.x = 2*p.x;
    p.y = 2*p.y;
}
public static void main (String [ ] args) {
   Point p1 = new Point (1, 4);
   Point p2 = new Point (2, 3);
   p2.y = p1.x;
   p1 = p2;
   times2 (p2);
}
```

Your five-digit number: _____

Problem 2 (4 points)

Before you continue with this problem, read the document "Background on the String and StringBuilder classes".

Part a

A CS 61B student, running the following JUnit test case, was surprised that the assertEquals assertion failed. Explain why this happened.

```
void testMatch ( ) {
   StringBuilder s1 = new StringBuilder ("ABC");
   StringBuilder s2 = new StringBuilder ("ABC");
   assertEquals (s1, s2);
}
```

Part b

Can an equals method be coded and added to the JUnit test class so that the assertion succeeds? Briefly explain your answer.

Your five-digit number: _____

Problem 3 (10 points)

Before you continue with this problem, read the document "Background on the String and StringBuilder classes".

Part a

Fill in the blanks in the edit method whose framework appears below. The argument represents a line of text typed at a console window, and processes it as follows.

- The line is processed left to right, starting with the character at position 0.
- For most characters, processing doesn't change the line. There are two exceptions.
  - # (a "sharp" sign) represents the "backspace" key. The text character immediately preceding the #, along with the # itself, are to be deleted. Backspaces that back up past the beginning of the line are to be ignored.
  - % (percent sign) represents an "escape". The next character, no matter what it is, is to be accepted as a character without special meaning. (You may assume that a stand-alone % will not occur as the last character in the line.) To put # or % into the line, one would type %# or %%.

Here's an example. If the argument is

```
%%NOW A\LAY#A##I##IS THE TA#IME ...%%
```

the modified string will be

```
%NOW IS THE TIME ...%
```

```
public static void edit (StringBuilder line) {
   for (int k=0; k<line.length(); ) {
      if (line.charAt(k)=='#') {

      } else if (line.charAt(k)=='%') {

      } else {

      }
   }
}
```

Problem 3, continued

Part b

Write a JUnit test case that verifies that the result of editing a line containing "ABC" is "ABC".

```
public void testNoSpecialChars ( ) {

}
```

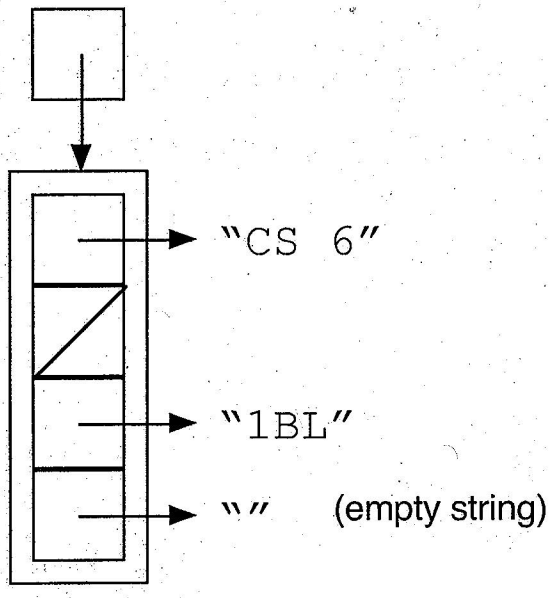
Your five-digit number: _____

Problem 4 (11 points)

Part a

Consider a class that represents a collection of String objects, and suppose that the collection is implemented as an ArrayList<String> named myStrings. On the next page is most of the code for an iterator of the characters in the strings. For example, if the array contains strings as shown below, successive calls to nextChar should return the characters in the sequence 'C', 'S', ' ', '6', '1', 'B', 'L'. Your first task is to complete the code by supplying the body of the advance method.

myStrings



Problem 4, continued

Iterator code, maintaining the invariant that between calls to nextChar, either a character to be returned is at myStrings.get (arrayIndex).charAt (stringIndex) or hasMoreChars is false.

```
// position of one of the strings in myStrings
private int arrayIndex;

// position of one of the chars in myStrings.get (arrayIndex)
private int stringIndex;

public void initIterator ( ) {
   arrayIndex = 0;
   stringIndex = 0;
   advance ( );
}

public boolean hasMoreChars ( ) {
   return arrayIndex < myStrings.size ( );
}

public char nextChar ( ) {
   char rtn = myStrings.get (arrayIndex).charAt (stringIndex);
   stringIndex++;
   advance ( );
   return rtn;
}

private void advance ( ) {
// You write this code so that it maintains the invariant given above.
```

Problem 4, continued

Part b

Fill in the table below with three test arrays, all of length four, that together with the example provide as much evidence as possible of the correctness of the iterator methods. Also indicate, for each test array, what unique information it provides about the iterator methods' correctness.

Element of myStrings				Evidence provided
0	1	2	3	
`"CS 6"`	`null`	`"1BL"`	`""`	(the example) contains both a null and an empty string, with the latter at the end of the array

Problem 4, continued

Part c

Provide a boolean isOK method, to be called after every call to initIterator, hasMoreChars, and nextChar, that checks that the iterator methods and state variables satisfy the invariant described in part a.

Your five-digit number: _____

Problem 5 (3 points)

The following class represents a year in this century as an int between 2000 and 2099, inclusive. The constructor successfully initializes a year from a legal string, but completely ignores the possibility of an illegal argument.

```
public class Year {

   private int myYear;

   public Year (String s) {
      Scanner intScanner = new Scanner (s);
      myYear = intScanner.nextInt ( );
   }
}
```

Rewrite the constructor to detect *any* error in the argument other than leading or trailing white space, and throw IllegalArgumentException with an informative message. (A full-credit solution will have at least four different informative messages.) Most of the error detection should be handled by Scanner.nextInt, which treats its String argument as a sequence of "tokens"—sequences of non-whitespace characters—that get returned one by one similar to the way an iterator works. Scanner.nextInt tries to interpret the next token as an int, and may throw the following exceptions:

- NullPointerException if the argument string is null;
- NoSuchElementException if no more non-whitespace characters appear in the string;
- InputMismatchException if the next token isn't an integer.

You may also find the Scanner.hasNext method useful. It returns true when at least one token remains in the string.

```
public Year (String s) {
   Scanner intScanner = new Scanner (s);
```