1. (6 pts) To the right of the line, draw a picture showing everything (variables/objects/etc.) after Java executes the following statements (do this on scratch paper first, and then redraw everything nicely here). Also, after each word returns (below the code) indicate what value each of the operators/methods returns.

```java
String s1 = new String("abc");
String s2 = s1;
String s3 = new String("ABC");
String s4 = s2;
s1 = new String("abc");
```

- `s2 == s4` returns
- `s2 == s1` returns
- `s2 == s3` returns
- `s1.equals(s3)` returns
- `s1.equals(s2)` returns
- `s1.compareTo(s3)` returns
- `s1.compareTo(s2)` returns

2. (6 pts) Complete the code fragment below. It prints the token in `allwords` that directly precedes the first occurrence of `word`. E.g., if `allwords` stores "ab lm pq xyz pq vl" and `word` stores "pq" it prints "lm" only; if `word` (a) appears first in `allwords`, print "*none*" (b) does not appear in `allwords`, print nothing. Use the `StringTokenizer` class in your code to help solve this problem (and any other Java statements we know). Hint: Think how you would solve this problem if someone handed you a sequence of cards that each contained a word.

```java
String allwords = Prompt.forString("Enter list of all words separated by a space");
String word = Prompt.forString("Enter one word");
String previous = "*none*";
```
3a. (5 pts) Examine the definition of the class Quiz5 below. (a) To the right of each member, circle any abbreviation that describes it (in some cases none apply; in others more than one apply). The key is: as follows: \( C \) = Constructor; \( M \) = Method; \( F \) = Field; \( O \) = Overloaded. At the end of each line that defines a method, write either \( a \) or \( m \), depending on whether you would classify that method most likely as an accessor or a mutator.

```
public class Quiz5 {
    public Quiz5 () {...} C M F O
    public Quiz5 (int l, int m, int r) {...} C M F O
    public int getMiddle() {...} C M F O
    private int getRight () {...} C M F O
    public void incAll (int delta) {...} C M F O
    public void incAll (Quiz5 q) {...} C M F O
    public void switchAll() {...} C M F O
    public void switchAll(Quiz5 q) {...} C M F O
    public int add () {...} C M F O
    public static int add(int l, int m, int r) {...} C M F O
    public static int add(Quiz5 q) {...} C M F O
    public int left; C M F O
    private int middle; C M F O
    private int right; C M F O
    public static int newCount = 0; C M F O
}
```

3b. (8 pts) Continue using the members of the Quiz5 class defined above in this part of the question. Write a legal Java statement to meet each of the following requirements (or write \textbf{impossible}). Pay very close attention to the access modifiers (public, private, and static) appearing before each member of the class, because they will help differentiate correct from incorrect solutions: different access modifiers require different syntax. Assume, wherever useful, that the variables \( x \) and \( y \) have been declared and initialized as follows: Quiz5 \( x = \text{new Quiz5}() \), \( y = \text{new Quiz5} \ (1,2,3) \); Sometimes you will need to supply literal arguments to these methods; use any, but ensure that they are of the correct type.

(a) Print the value returned by any call to \textbf{getMiddle}:

(b) Print the value returned by calling \textbf{getRight}:

(c) Call \textbf{incAll} (using the “one object” form):

(d) Call \textbf{incAll} (using the “two objects” form):

(e) Call either form of \textbf{switchAll}:

(f) Print the value returned by calling the first form of \textbf{add}

(g) Print the value returned by calling the second form of \textbf{add}:

(h) Print the value of any \textbf{left} instance variable without calling \textbf{Quiz5} methods:

(i) Print the value of \textbf{newCount} without calling \textbf{Quiz5} methods: