Question 1

25 minutes to complete, 5 minutes to upload.

Many encoded strings contain delimiters. A delimiter is a non-empty string that acts as a boundary between different parts of a larger string. The delimiters involved in this question occur in pairs that must be balanced, with each pair having an open delimiter and a close delimiter. There will be only one type of delimiter for each string. The following are examples of delimiters.

Example 1
Expressions in mathematics use open parentheses "(" and close parentheses ")" as delimiters. For each open parenthesis, there must be a matching close parenthesis.

- (x + y) * 5 is a valid mathematical expression.
- (x + (y) is NOT a valid mathematical expression because there are more open delimiters than close delimiters.

Example 2
HTML uses <B> and </B> as delimiters. For each open delimiter <B>, there must be a matching close delimiter </B>.

- `<B> Make this text bold </B>` is valid HTML.
- `<B> Make this text bold </UB>` is NOT valid HTML because there is one open delimiter and no matching close delimiter.

In this question, you will write two methods in the following Delimiters class.

```
public class Delimiters
{
	/** Instance variables. */
	private String openDel;
	private String closeDel;
	private String[] tokens;

	/** Constructs a Delimiters object where open is the open delimiter and close is
	*   the close delimiter.
	*   Precondition: open and close are non-empty strings.
	*/
	public Delimiters(String o, String c, String[] t)
	{
		openDel = o;
		closeDel = c;
		tokens = t;
	}

	/** Returns an ArrayList of integers, as described in part (a). */
	public ArrayList<Integer> precedenceRank()
	{ /* to be implemented in part (a) */ }

	/** Returns list of one or more token strings, as described in part (b). */
	public ArrayList<String> evaluateFirst()
	{ /* to be implemented in part (b) */ }

	// There may be instance variables, constructors, and methods that are not shown.
}
```

(a) A string containing text and possibly delimiters has been split into tokens and stored in String[] tokens. Each token is either an open delimiter, a close delimiter, or a substring that is not a delimiter. You will write the method precedenceRank, which returns an ArrayList containing integers indicating the level of nesting of each token. Whenever the token openDel is encountered, the count increases by 1, and whenever closeDel is encountered, the count decreases by 1.
The following examples show the contents of an ArrayList returned by getDelimitersList for different open and close delimiters and different tokens arrays.

Example 1

```
openDel: "("
closeDel: ")"
```

tokens	"a+"	"("	"("	"x+y"	")"	"*b"	")"
precedenceRank	0	1	2	2	1	1	0

Example 2

```
openDel: "<q>"
closeDel: "</q>"
```

tokens	"a"	"<q>"	"b"	"</q>"	"c"	"<q>"	"d"	"</q>"
precedenceRank	0	1	1	0	0	1	1	0

Complete the method precedenceRank:

```
/** Returns an ArrayList of delimiters from the array tokens, as described in part (a). */
public ArrayList<Integer> precedenceRank()
```

(b) Write the method evaluateFirst, which returns a list of one or more tokens (but not delimiter tokens) that have the highest precedence, which are the token(s) that should be evaluated before the other tokens.

Example 1

```
openDel: "("
closeDel: ")"
```

tokens	"a+"	"("	"("	"x+y"	")"	"*b"	")"
precedenceRank	0	1	2	2	1	1	0

evaluateFirst	"x + y"

Example 2

```
openDel: "<q>"
closeDel: "</q>"
```

tokens	"aa"	"<q>"	"bb"	"</q>"	"cc"	"<q>"	"dd"	"</q>"
precedenceRank	0	1	1	0	0	1	1	0

evaluateFirst	"bb"	"dd"

Complete method evaluateFirst below.

```
/** Returns list of one or more token strings, as described in part (b). */
public ArrayList<String> evaluateFirst()
```