

Part I – Multiple Choice

The answers to the multiple choice exercises are: CDCBEDADAABAADCBBCEE

Part II – Free Response

1. Write a method named `remove` as started below. The method finds the first occurrence of a specified `String` named `word` in a given `String` named `text` and returns `text` with `word` removed. There should only be one space in the position where `word` is removed. In other words, there should not be any occurrences of two consecutive blank spaces. If `word` is not found, the method returns `text` unchanged.

```
// precondition: text is a non-empty string that consists of words separated by single blank spaces
// postcondition: if word is found in text, its first occurrence is removed from text and text is then
//                returned; otherwise text is returned unchanged
```

```
public static String remove(String text, String word)
{
    if (text.indexOf(word) != -1)
    {
        String firstPart = text.substring(0, text.indexOf(word));
        String lastPart = text.substring(text.indexOf(word) + word.length());
        return firstPart + " " + lastPart;
    }
    return text;
}
```

2. Assume that you are given the following `Sentence` class which has a class invariant that `myWords` will always contain a set of English words that are separated by single blank spaces. Each word in `myWords` only contains letters. That is, there are no punctuation symbols (such as periods, question marks, or even hyphens) or digits within `myWords`. There is no leading blank space in front of the first word in `myWords` and there is no trailing blank space behind the last word in `myWords`. You are also guaranteed as a class invariant that `myWords` will always contain at least 2 or more words.

```
public class Sentence
{
    private String myWords;

    public Sentence()
    {
        myWords = "I love Java programming";
    }

    public Sentence(String words)
    {
        myWords = words;
    }

    public String getWords()
    {
        return myWords;
    }

    // a
    // postcondition: the number of separate words in myWords is returned
    public int numWords()
    {
        int count = 0;
```

```

    // counting the total number of spaces and adding one
    for (int i = 0; i < myWords.length(); i++)
    {
        if (myWords.substring(i, i+1).equals(" "))
        {
            count++;
        }
    }

    return count + 1;
}

// b
// postcondition: the number of separate letters in myWords is returned
public int numLetters()
{
    // subtracting the # of spaces from the total number of characters
    return myWords.length() - (numWords() - 1);
}

// c
// precondition: key will not be a null string and it will be a word
// that only contains letters
// postcondition: if the String key is found anywhere within myWords,
// the boolean value true will be returned;
// otherwise false will be returned.
public boolean find(String key)
{
    if (myWords.indexOf(key) >= 0)
    {
        return true;
    }

    return false;
}

// d
// precondition: num > 0 and num will be less than or equal to
// the number of words in myWords
// postcondition: return the numth word in myWords where the very
// first word would be returned if num is 1
public String getWord(int num)
{
    if (num == 1)
    {
        return myWords.substring(0, myWords.indexOf(" "));
    }

    int start = myWords.indexOf(" ") + 1;
    int end = myWords.length();
    int count = 1;

    for (int i = myWords.indexOf(" ") + 1; i < myWords.length() - 1; i++)
    {
        if (myWords.substring(i, i + 1).equals(" "))
        {
            count++;
        }
    }
}

```

```

        if (count == num - 1)
        {
            start = i + 1;
        }

        if (count == num)
        {
            end = i;
            break;
        }
    }

    return myWords.substring(start, end);
}

// e
// precondition: return the boolean value true if any word
// occurs twice or more in myWords
public boolean containsDouble()
{
    for (int i = 1; i <= numWords(); i++)
    {
        for (int j = i + 1; j <= numWords(); j++)
        {
            if (getWord(j).equals(getWord(i)))
            {
                return true;
            }
        }
    }

    return false;
}

// f
// precondition: return the boolean value true if any word
// ends with the letter lowercase 's'
public boolean containsWordThatEndsWithS()
{
    for (int i = 1; i <= numWords(); i++)
    {
        if (getWord(i).substring(getWord(i).length()-1).equals("s"))
        {
            return true;
        }
    }

    return false;
}

}

/*

public class SentenceTest
{
    public static void main(String[] args)
    {
        Sentence test = new Sentence("I love Macintosh computers NOT");
    }
}

```

```
System.out.println("myWords: " + test.getWords());
System.out.println("A. numWords: " + test.numWords());
System.out.println("B. numLetters: " + test.numLetters());
System.out.println("C. find(love): " + test.find("love"));
System.out.println("D. getWord(3): " + test.getWord(3));
System.out.println("E. containsDouble: " + test.containsDouble());
System.out.println("F. containsWordThatEndsWithS: " +
    test.containsWordThatEndsWithS());
```

```
}
```

```
}
```

```
*/
```