CS1020 SIT-IN LAB 04 A – WORD SCRAMBLE

Semester 2 AY2014/2015

Advice

1. Understand the problem thoroughly. Ask the invigilator when unsure.
2. Check out the given input and output files, and use the "diff" command to compare your own output with the expected output.
3. Make sure your code can be compiled ***AT ALL TIMES!***
4. Note that code that is commented out will **not** be looked at and graded.

Important notes:

1. You are to use **recursion**. No mark will be awarded for correctness even if your program passes the test data but does not use recursion in a correct and meaningful manner. Using recursion only for peripheral tasks (eg: reading the input) does not count as meaningful usage.
2. You are **NOT** to use any String (or its variants) API methods to solve the problem. You may, however, use the **toCharArray()** method to convert a string into a character array.
3. Note that there are **4** test cases for part 1 and **6** test cases for part 2.

Problem Description

Part 1

We want to design a word finding game. The input will be an array of characters and a search word. A search word may consist of a single character or multiple characters. The program needs to find the search word in the character array. If the search word is found in more than one places in the character array, the first one encountered will be reported.

The search is case-sensitive. You may assume the array and the search word contain only letters.

Input

The first line of input is an integer 1 indicating part 1. The second line contains the characters to be read into the array. The search word is given in the third line.

Two examples are shown below. (Note that comments on the right are for explanatory purpose and they do not appear in the input.)

Input #1

```
1
breadtalk
talk
```

Input #2

```
1
breadtalk
real
```

The test case is for part 1.
Input for the character array.
Input for the search word.

Output

The program should output either **Found at position *n*** where *n* is the starting position of the word found in the character array, or **Not found** if the word is not present in the character array.

For the above two examples the outputs are:

Output #1

```
Found at position 5
```

Output #2

```
Not found
```

Part 2

Consider the same word finding problem but now the characters of the word can be separated in the array by other characters in between them. However, relative order among the characters of the word needs to be maintained i.e. instead of a substring we want to search for a subsequence.

For this part, there can be more than one search word. As in part 1, the search is case-sensitive.

Input

The first line of input is an integer 2 indicating part 2. The next line contains the characters to be read into the array. The next line is a positive integer *N* indicating the number of search words. Subsequent *N* lines contain the search words.

An example is shown below. (Note that comments on the right are for explanatory purpose and they do not appear in the input.)

```
2
breadtalk
3
real
bet
tar
```

The test case is for part 2.
Input for the character array.
There are 3 search words.
First search word.
Second search word.
Third search word.

Output

For each of the search words, the program should output either **Found at position *n*** where *n* is the starting position of the word found in the character array, or **Not found** if the word is not present in the character array.

For the above input the output is:

```
Found at position 1
Found at position 0
Not found
```

Skeleton program

You are provided with a skeleton program **WordScramble.java**. Do not create any other file and do not rename this file or the public class of this file. You may change the skeleton code, add methods and classes any way you deem fit.

```
import java.util.Scanner;

public class WordScramble {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        WordScramble ws = new WordScramble();
        ws.play(sc);
    }

  /**
   * Reads the input and calls the appropriate method to search the word
   * @param sc : Scanner object to be used for reading
   */
  public void play(Scanner sc) {
      int part = Integer.parseInt(sc.nextLine());
      char[] array = sc.next().toCharArray();

      if (part == 1) {
          char[] word = sc.next().toCharArray();
          findWordInArray(array, word, part);
      } else if (part == 2) {
          int numberOfWords = sc.nextInt();
          for (int i =0;i < numberOfWords; i++) {
              char[] word = sc.next().toCharArray();
              findWordInArray(array, word, part);
          }
      }
  }

}
```

-- **Have fun** !!! ☺ --