CS1020 Sit-in Lab 04 B - String Replacement

Semester 2 AY2013/2014

Requirements

Use recursion to solve the tasks in this sit-in lab. NO for/while/do..while loops should be used in your code. **No correctness marks will be given for iterative solutions. Correctness marks will be halved for code that use for/while/do..while loops within the recursive solutions.**

You can use Java String, **but you are not allowed to use any String methods except for charAt, substring, length, isEmpty, concat. Using any other String method will result in 0 correctness marks.**

Problem Description - Part 1 (4 test cases/28 marks)

Given a non-empty **string** A composed of alphanumeric characters (lowercase a..z and digits 0..9), and an alphanumeric **character** C, replace all occurrences of C in A with a single white space. A and C are all in lower case. Write a recursive method to do this. An example is given below

```
jxcbgakkja1kuigacglq73uguqew <-- string A
k <-- character C

jxcbgakkja1kuigacglq73uguqew <-- occurrences of k are as indicated
jxcbga  ja1 uigacglq73uguqew <-- each occurrence of k is replaced by a white space
```

Input

The 1st line in the input is the value 1. This indicates the input is for part 1. The next line is the string A. The line following that is the character C. An example input is given below:

```
1  <-- indicates this is input for part 1
jxcbgakkja1kuigacglq73uguqew  <-- string A
k  <-- character C
```

Output

Print out the replaced string. Example output for the given input above is shown below:

```
jxcbga  ja1 uigacglq73uguqew
```

Skeleton Program

Your program is to be named **StrReplacement.java** (do not change this name). A skeleton program is provided, but you can change it any way or add new class(es) and method(s) as you deem fit. **Write all your code in StrReplacement.java, do not create new files!** The skeleton program is as follows:

```
import java.util.*;

/* class representing the String Replacement problem */
public class StrReplacement {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

    }
}
```

Testing your program

The following **sample** input and output files are in your plab account:

```
StrReplacementTC1.in, StrReplacementTC2.in, StrReplacementTC3.in
StrReplacementTC1.out, StrReplacementTC2.out, StrReplacementTC3.out
```

StrReplacementTC1.in is the input test case for **part 1**, StrReplacementTC2.in is the input test case for **part 2** and StrReplacementTC3.in is the input test case for **part 3**. StrReplacementTC1.out, StrReplacementTC2.out, and StrReplacementTC3.out are the expected output for the respective test cases.
After you have compiled your program, to test it with say StrReplacementTC1.in, you type:

```
java StrReplacement < StrReplacementTC1.in
```

Please complete part 1 before starting on part 2 !

Problem Description - Part 2 (4 test cases/28 marks)

Given a non-empty **string** A and non-empty **string** B composed of alphanumeric characters (lowercase a..z and digits 0..9), where length of B $\leq$ length of A, replace all occurrences of B in A with a single white space. Assume that A and B are all in lower case. Write a recursive method to do this. An example is given below

```
jxcbgakkja1kuigacglq73uguqew <-- string A
ga <-- string B

jxcbgakkja1kuigacglq73uguqew <-- occurrences of ga are as indicated
jxcb kkja1kui cglq73uguqew <-- each occurrence of ga is replaced by a white space
```

Input

The 1st line in the input is the value 2. This indicates the input is for part 2. The next line is the string A. The line following that is the string B. An example input is given below:

```
2  <-- indicates this is input for part 2
jxcbgakkja1kuigacglq73uguqew  <-- string A
ga  <-- string B
```

Output

Same as before. Example output for the given input above is shown below:

```
jxcb kkja1kui cglq73uguqew
```

Please complete part 2 before starting on part 3 !

Problem Description - Part 3 (2 test cases/14 marks)

Given a non-empty **string** A and non-empty **string** B composed of alphanumeric characters (lowercase a..z and digits 0..9), where length of B $\leq$ length of A, find the **longest common substring** between A and B. Next, replace all occurrences of that longest common substring in A with a single white space. Assume that A and B are all in lower case. You can also assume that the longest common substring between A and B **will always be unique, but not necessarily non-empty**. Write a recursive method to do this. An example is given below

```
88cdegcgwcdeighkcdedkwxhkd <-- string A
abcdef                     <-- string B

cde <-- longest common substring between A & B

88cdegcgwcdeighkcdedkwxhkd <-- occurrences of cde in A are as indicated
88 gcgw ighk dkwxhkd <-- each occurrence of cde is replaced by a white space
```

Input

The 1st line in the input is the value 3. This indicates the input is for part 3. The next line is the string A. The line following that is the string B. An example input is given below:

```
3  <-- indicates this is input for part 3
88cdegcgwcdeighkcdedkwxhkd  <-- string A
abcdef  <-- string B
```

Output

Same as before. Example output for the given input above is shown below:

```
88 gcgw ighk dkwxhkd
```

Grading Scheme

1. Program correctness = 70 marks, Design = 20 marks, Programming = 10 marks
2. No marks awarded if the program does not compile.
3. Marks will be deducted if **student particulars and program description** are not filled up in the top portion of the source code.
4. There are 10 test cases. Each test case is worth 7 marks.
5. Correctness marks will be not be given or will be halved as indicated in the requirements.
6. Things to look out for under design

   (a) correct usage of programming constructs (eg use correct type for variables)
   (b) Not overly complicated logic
   (c) No redundant logic

7. Things to look out for under programming style
   (a) Meaningful comments
   (b) Pre-condition (if appropriate) should be present in the comment above a recursive method.
   (c) Proper indentation
   (d) Meaningful identifiers