COS 126 Programming Exam 1 Fall 2019

Instructions. You will have 50 minutes to create and submit one program and a `readme` file. Download the project zip file, which includes all the files you will need, from the **`Exams`** page.

Resources. You may use your book, your notes, your code from programming assignments and precepts, the code on the COS 126 course website, the booksite, and you may read Ed Discussion. No form of communication is permitted (e.g., talking, texting, etc.) during the exam, except with course staff.

Submissions. Submit your work using the `Submit!` link on the **`Exams`** page.

Grading. Your program will be graded on correctness, clarity (including comments), design, and efficiency. You will lose a substantial number of points if your program does not compile or if it crashes on typical inputs.

Discussing this exam. Due to travel for extracurriculars and sports, some of your peers will take this exam next week. Do not discuss exam contents with anyone who has not taken the exam.

This piece of paper. In addition to submitting your code electronically, you *must* turn in this piece of paper. Fill in the information below, then transcribe and sign the Honor Code pledge. You may do so now.

NAME: _____

LOGIN: _____

PRECEPT: _____

EXAM ROOM: _____

"I pledge my honor that I have not violated the Honor Code during this examination."

_____

[*copy the pledge onto this line*]

_____

[*signature*]

Programming Exam 1: String similarity

Part 1 (7 points). Given a string `s`, the Java code `s.charAt(0)` gives the first character of `s`, the code `s.charAt(1)` gives the second character of `s`, and so forth. Define the *similarity* of two strings `a` and `b` as the number of indices `i` for which `a.charAt(i) == b.charAt(i)`. For example, the similarity of `truth` and `true` is 3 and the similarity of `nearly` and `neatly` is 5. Your task is to write a program that finds a string in standard input whose similarity with the string on the command line is maximal.

```
% more BestMatch.java
public class BestMatch
{
    public static int similarity(String a, String b)
    {
       // YOUR CODE HERE
    }

    public static void main(String[] args)
    {
       StdOut.println(similarity("truth", "true");
       StdOut.println(similarity("nearly", "neatly");
       // REPLACE WITH YOUR CODE
       // AFTER GETTING similarity() WORKING
    }
}
```

Starting point. Begin with the template shown at right (the file `BestMatch.Java` from the project zip file on from the **`Exams`** page).

First step. Fill in the code for the static method `similarity()` to calculate the similarity of the two given strings. For your convenience in debugging `similarity()`, the template includes a few calls to it. *Important tip:* As your first statement, use the code `int N = Math.min(a.length(), b.length());` This will help you avoid the run-time error of calling `s.charAt(i)` for `i` greater than the length of the string.

Second step. Replace the two lines of code in `main()` with code that calculates the similarity of each string on standard input with the command-line argument (by calling `similarity()`), keeps track of the *largest* value seen so far, and prints the result.

Input. Take a string from the command line and a sequence of strings from standard input (use `StdIn.readString()` to read them).

Output. Print the maximal similarity value found, then the first string on standard input having that value. If the value is 0, just print 0.

Example. For the file `tiny.txt` shown at right, your program must behave as follows:

```
% java-introcs BestMatch actactg < tiny.txt
5 actgcta

% java-introcs BestMatch aaaaaaa < tiny.txt
5 acacaaactt
```

```
% more tiny.txt
actgcta
ctgactggaa
acacaaactt
acacaatgtgtg
actacat
```

Submit `BestMatch.java` using the `Submit!` link on the **`Exams`** page

Do not attempt Part 2 until you have submitted your solution to Part 1.

You cannot get credit for Part 2 unless your solution to Part 1 is correct.

There is no partial credit for Part 2.

Part 2 (0.5 points). Define the *sliding similarity* of two strings `a` and `b` as the maximum, over all values of a starting point `start`, of the number of indices `i` for which `a.charAt(i) == b.charAt(i+start)`. For example, the sliding similarity of `truth` and `untrue` is 3 and the sliding similarity of `aaaaa` and `acacaaactt` is 4 (see illustration below). Copy your solution `BestMatch.java` to make a new program `BestSlidingMatch.java` and then modify it to make a program that can find a string in standard input whose sliding similarity with the string on the command line is maximal.

start position	a c a c a a a c t t	*character matches*
0	**a a a a a**	3
1	a **a a a a**	3
2	**a** a **a a a**	4
3	a **a a a** a	3
4	**a a a** a a	3
5	**a a** a a a	2
	max value	**4**

Submit `BestSlidingMatch.java` using the `Submit!` link on the **`Exams`** page.