Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Programming Assignment 2 (15%) CSI2110/CSI2510 **2021**

Recommended deadline: Nov 29, 11:59PM (2 out of 60 early bonus marks)

Accepted deadlines : Dec 1, 11:59PM (1 out of 60 early bonus mark), Dec 3, 11:59 (no early bonus)

Late assignment policy : ***1min-24hs late are accepted with 30%off; no assignments accepted after 24hs late.***

Introduction

In this assignment, you will solve problems used for practice for ACM programming competitions. The focus here is for you to solve the problem on your own, and not inspect any previous solutions to this problem. Your source code will be inspected by the teaching assistants and programs will be screened for plagiarism; the best way to avoid plagiarism is not to even look at any piece of code that solves the problem.

For this assignment, you will be submitting your code to an online judge which tests your code and gives a verdict. To receive the "accepted" verdict, your code needs to correctly solve their battery of tests within the allotted maximum time.

The input and output format for the online judge is strict and you should observe its rules otherwise your program will not run properly.

For submission specifications see:

https://onlinejudge.org/index.php?option=com_content&task=view&id=15&Itemid=30

"The program reads the test input from the standard input (stdin) and places the results in the standard output (stdout)."

"The Java programs submitted must be in a single source code (not .class) file. Nevertheless, you can add as many classes as you need in this file. All the classes in this file must not be within any package." (...)

"All programs must begin in a static main method in a Main class.

Do not use public classes: even Main must be non public to avoid compile error.

Use buffered I/O to avoid time limit exceeded due to excesive flushing.

As a reference, we provide a sample Java code"

For veredict information see:

https://onlinejudge.org/index.php?option=com_content&task=view&id=16&Itemid=31

There are two types of efficiency considerations: complexity issues (big Oh) and other programming considerations that can affect running time (e.g. things that affect the constant in front ot the big-Oh such as the class you use to read the input, reducing unecessary tasks/steps, etc). Focus on reduced complexity (big-Oh), but also pay attention to other advice (such as buffered I/O) and making the code simple and fast for further efficiency improvements.

Problem 1: Document analyser

Problem description in appendix, UVa ID: 11860 *obtained at* https://onlinejudge.org/

Efficiency considerations

To detect duplicate words efficiently, we advise you to use hash tables. Assuming that your hash table allows for search in constant time, your algorithm **should run in time *O(n).***

Problem 2: Continents

Problem description in appendix, UVa ID: 11094 *obtained at* https://onlinejudge.org/

Requirements:

You will solve two variations of this problem:

Part A) The problem as stated in the online judge handout.

Part B) A variation of the problem as specied below (same input as online judge but different output according to our specifications).

Using breath-first search (BFS) on a graph:

You can see that both part A and B can be solved by finding connected components on a graph based on the problem input. As you have learned in class, connected components can be found via depth-first search or breadth-first search. However, the answer for part B depends on unit distances on a connected component, which can be obtained using breadth-first search.

Note that you do not need to explicitly create a complete data structure for a general graph to apply breadth-first search; the BFS traversal can be done by using the notions of "adjacency" known for this specific problem.

Your tasks:

Part 1: Problem 1 - Document Analyser

- Implement your code as efficiently as possible (inefficient solutions may not pass the online judge).
- Debug and test using the test cases provided with this assignment.
- Test using the online judge and record the results by taking screen shots of the veridic for your submissions (see sample table in appendix).

Part 2: Problem 2 - Continents

For this part, a single code must handle both the original problem (Part A) and the variation (Part B).

Part A will be run in the command line by using no command-line arguments, as follows:

```
java Main <input1.txt  >output1.txt
```

"`<input1.txt`" redirects standard input from file and "`>output1.txt`" redirects standard output to file.

Part B will be run with command-line argument "`PartB`", as follows:

```
java Main PartB <input1.txt  >output1.txt
```

- Implement your code as efficiently as possible (inefficient solutions may not pass the online judge).
- Debug and test using the test cases provided with this assignment.
- For partA, test using the online judge and record the results by taking screen shots of the veridic for your submissions.
- For partB, test using the files provided and provide files containing each of the outputs with corresponding standard names by appending suffix "out" to file name (e.g for input "sample1.txt" the output is "sample1out.txt")

Part 3: Report

Write a short report about your findings, including the following sections and information:

1) **Report on the status of your code for each part:**

- For Part 1: Summary of code status regading correctness, report on known bugs, indicate if it passed all our tests and if it was accepted by the online judge; provide online judge screen shots and report on running time for the online judge on this test.
- Status of Part 2A: Summary of code status regading correctness, report on known bugs, indicating if it passed all our tests and if it was accepted by the online judge; provide online judge screen shots and report on running time for the online judge on this test.
- Status of Part 2B: Summary of code status regading correctness, report on known bugs, indicating if it passed all our tests.

2) Report on program development

Give any details used in your design and describe any improvements in performance you obtained in this process.

SUBMISSION REQUIREMENTS

- Your submission must be in a zipped folder called p2<student number>. In the main directory of this folder we must have:
  - subdirectory `Part1` containing: Main.java, your solution for the first problem; input and output files.
  - subdirectory `Part 2` containing: Main.java, your solution for the second problem; input and output files.
  - A file called report.pdf or report.doc containing your complete report.
  - Optional readme.txt file explaining the status of your code, known bugs, success in various tests, or anything your wish to tell the marker, etc

<u>Marking Scheme (60 marks, 100%)</u>

Part 1:

- Code design and quality for Document Analyser (12 marks, 20%)
- Correctness for our testing (6 marks, 10%)
- Online judge acceptance (6 marks, 10%)

Part 2:

- Code design and quality for Continents - part A and part B (12 marks, 20%)
- Correctness for our testing part A (6 marks, 10%)
- Online judge acceptance part A (6 marks, 10%)
- Correctness for our testing part B (6 marks, 10%)

Part 3:

- Report (6 marks 10%)

Note: Complete assignments submitted by the due date that are deemed of exceptional quality due to ingenuity and efficiency clearly demonstrated in the report and experimental results may qualify for 6 bonus points, 10%, (quality to be decided on a comparative/competitive basis); email your professor if you think your assignment may qualify for bonus.

APPENDIX:

UVa Online Judge

11860 Document Analyzer

You work in a leading software development company. As you are great in coding, most of the critical tasks are allotted for you. You like the challenge and you feel excited to solve those problems.

Recently your company is developing a project named *Document Analyzer*. In this project you are assigned a task; of course a critical task. The task is that you are given a document consisting of lowercase letters, numbers and punctuations. You have to analyze the document and separate the words first. Words are consecutive sequences of lower case letters. After listing the words, in the order same as they occurred in the document, you have to number them from 1, 2, ..., n. After that you have to find the range p and q ($p \leq q$) such that all kinds of words occur between p and q (inclusive). If there are multiple such solutions you have to find the one where the difference of p and q is smallest. If still there is a tie, then find the solution where p is smallest.

Since you do have other works to do, you have to solve this task within 5 hours.

Input

First line of input will contain T (≤ 20) denoting number of documents.

Each document will be denoted by one or more lines, each line having no more than 150 characters. A document will contain either lowercase letters or numbers or punctuations. The last line of a document will contain the word 'END' which is of course not the part of the document. You can assume that a document will contain between 1 and 10^5 words (inclusive).

Output

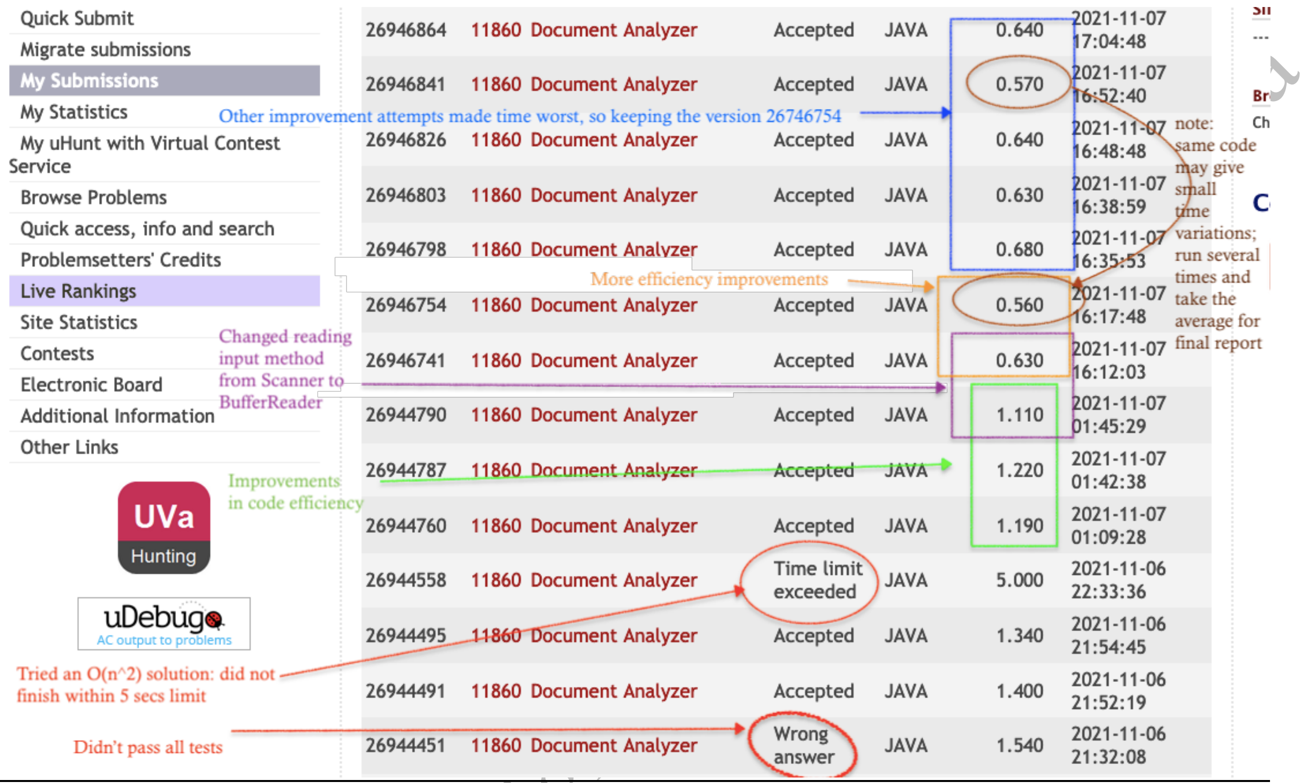
For each document, print the document number first. After that, print p and q as described above. For exact formatting, see the samples.

Sample Input

```
3
1. a case is a case,
2. case is not a case~
END
a b c d e
END
a@#$a^%a a a
b b----b b++12b
END
```

Sample Output

```
Document 1: 6 9
Document 2: 1 5
Document 3: 5 6
```



26946864	11860 Document Analyzer	Accepted	JAVA	0.640	2021-11-07 17:04:48
26946841	11860 Document Analyzer	Accepted	JAVA	0.570	2021-11-07 16:52:40
26946826	11860 Document Analyzer	Accepted	JAVA	0.640	2021-11-07 16:48:48
26946803	11860 Document Analyzer	Accepted	JAVA	0.630	2021-11-07 16:38:59
26946798	11860 Document Analyzer	Accepted	JAVA	0.680	2021-11-07 16:35:53
26946754	11860 Document Analyzer	Accepted	JAVA	0.560	2021-11-07 16:17:48
26946741	11860 Document Analyzer	Accepted	JAVA	0.630	2021-11-07 16:12:03
26944790	11860 Document Analyzer	Accepted	JAVA	1.110	2021-11-07 01:45:29
26944787	11860 Document Analyzer	Accepted	JAVA	1.220	2021-11-07 01:42:38
26944760	11860 Document Analyzer	Accepted	JAVA	1.190	2021-11-07 01:09:28
26944558	11860 Document Analyzer	Time limit exceeded	JAVA	5.000	2021-11-06 22:33:36
26944495	11860 Document Analyzer	Accepted	JAVA	1.340	2021-11-06 21:54:45
26944491	11860 Document Analyzer	Accepted	JAVA	1.400	2021-11-06 21:52:19
26944451	11860 Document Analyzer	Wrong answer	JAVA	1.540	2021-11-06 21:32:08

Other improvement attempts made time worst, so keeping the version 26746754

note: same code may give small time variations; run several times and take the average for final report

More efficiency improvements

Changed reading input method from Scanner to BufferReader

Improvements in code efficiency

Tried an O(n^2) solution: did not finish within 5 secs limit

Didn't pass all tests

Table 1: Sample results of online judge for document analyser

UVa Online Judge

11094 Continents

Mijid the Great is the king of Dodars territory. He likes to travel between the cities in his territory and actually, you can never see him in the same city as where he was the day before. Therefore, he captured all territories of his continent! In spite of this fact, he has seen all cities of his territory so far and wants to capture another continent in order to have some choices to travel into new cities. Now, having the world map, he needs your help to find the biggest continent except the one in which he resides.

Maps are given as $M \times N$ tables, filled with at most two different letters denoting land and water regions. A continent is a set of connected land regions which is completely surrounded by water regions or the end of map. Two regions are assumed to be connected if they have an edge in common. The coordinates of top left region is (0,0) and bottom right region $(M-1, N-1)$. Region with coordinates $(x, N-1)$ should be assumed to have a common edge with region $(x, 0)$ for every x between 0 and $M-1$ (inclusive).

Input

There will be several test cases. Each test case contains two integers $M \leq 20$ and $N \leq 20$ in the first line denoting the number of rows and columns in the map respectively. Next, there will be M lines of exactly N characters representing the map. Finally in the last line there would be two integers $0 \leq X < M$ and $0 \leq Y < N$, the coordinates of the region in which Mijid the Great currently stays. There will one blank line after each test case.

Output

For each test case, output a line containing an integer that is the number of regions in the biggest continent that Mijid the Great can capture.

Sample Input

```
5 5
wwwww
wwllw
wwwww
wllww
wwwww
1 3
```

Sample Output

```
2
```

Continents - variation (Problem 2 - part B)

This variation follows the input and specifications of problem "Continents".

In this variation, we consider the northwest point of each continent to be the first land region in the continent encountered by a top-to-bottom left-to-right reading of the matrix.

In the previous Sample Input, the northwest points of the 2 continents have coordinates: (1,2), (3,1).

Majid the Great's wife always practices violin in the northwest point of a continent; so Majid the Great is interested in knowing how far from this noisy spot he can possibly be in the new continent.

In this problem, we continue considering the biggest continent(s) that Majid the Great can capture, but want to know what is the distance of the farthest land region in this continent to the northwest point of this continent. If more than one continent ties as the biggest continent, we will record the data for the one with the largest such distance to the northwest point.

Output

For each test case, output a line containing two integers, the first integer (as before) is the number of regions in the biggest continent that Majid the Great can Capture, and the second integer is the largest distance from a land region in this continent to the northeast point in this continent. If more than one continent ties as the biggest continent, output the largest such distance encountered among them.

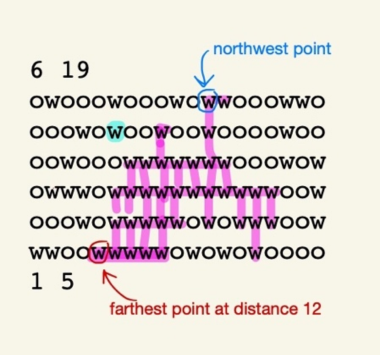
Sample Input

```
6 19
OWOOOWOOOWOWWOOOWWO
OOOWOWOOWOOWOOOOWOO
OOWOOOWWWWWWWOOOWOW
OWWWOWWWWWWWWWWWOOW
OOOWOWWWWWOWOWWWOOW
WWOOWWWWWOWOWOWOOOO
1 5
```

Sample Output

```
37 12
```

Justification for output:



```
6 19
OWOOOWOOOWOWWOOOWWO
OOOWOWOOWOOWOOOOWOO
OOWOOOWWWWWWWOOOWOW
OWWWOWWWWWWWWWWWOOW
OOOWOWWWWWOWOWWWOOW
WWOOWWWWWOWOWOWOOOO
1 5
```