**Java program:** Prob11.java

**Input File:** Prob11.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Camel Case is a naming style common in many programming languages. In Java, method and variable names typically start with a lowercase letter, with all subsequent words starting with a capital letter (example: startThread). Names of classes follow the same pattern, except that they start with a capital letter (example: BlueCar).

Your task is to write a program that creates or splits Camel Case variable, method, and class names.

**Program Input**

The file Prob11.in.txt will containa list of operations, object types, and words that you will need to operate on. The pattern will be: Operation;ObjectType;Words. Here is an explanation of the different elements of the input file:

* The operation will be either S (for split) or C (for combine).
* The Object Type will be M (for method), C (for class), or V (for variable)
* In the case of a split operation, the words will be a camel case method, class, or variable name that you need to split into a space-delimited list of words starting with lowercase letters.
* In the case of a combine operation, the words will be a space-delimited list of words starting with lowercase letters that you need to combine into the appropriate camel case string. Methods should end with an empty set of parentheses to differentiate them from variable names.

**Example Input:**

S;M;plasticCup()

C;V;mobile phone

C;C;coffee machine

S;C;LargeSoftwareBook

C;M;white sheet of paper

S;V;pictureFrame

**Program Output**

Your program’s output should display either the space-delimited list of words (in the case of a split operation) or the appropriate camel case string (in the case of a combine operation).

**Example Output:**

plastic cup

mobilePhone

CoffeeMachine

large software book

whiteSheetOfPaper()

picture frame

**Java program:** Prob12.java

**Input File:** Prob12.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is used heavily in the IT world to pass data around from system to system.

XML looks very much like HTML in that XML elements are encased in a set of less than and greater than signs. End tags are similar but have a slash at the beginning of the element name. Here is an example:

<MyElement>This is the text inside of a MyElement element!</MyElement>

XML tags can be named whatever you want, and they can be nested (which is what starts to unleash the true power of XML). Your job is to look through an XML file and list the element names that you find in order and the number of times that element name exists in the XML file. Remember - do not count the ending tags, only the starting ones!

**Program Input**

The file Prob12.in.txt will contain a snippet of XML. There will be no self-ending elements (i.e. <ThisIsSelfEnding />). Elements may or may not have any content.

**Example Input:**

<XML>

<MoviesILike>

<Movie>

<Title>Back to the Future</Title>

<Year>1982</Year>

</Movie>

<Movie>

<Title>Iron Man</Title>

</Movie>

</MoviesILike>

<MoviesIDoNotLike>

<Movie>

<Title>Steel Magnolias</Title>

</Movie>

<Movie>

<Title>Grease</Title>

</Movie>

</MoviesIDoNotLike>

</XML>

**Program Output**

Your program should print out the names of every XML element in the file in the order that you come across them, along with the number of times that element exists in the file separated by a space.

**Example Output:**

XML 1

MoviesILike 1

Movie 4

Title 4

Year 1

MoviesIDoNotLike 1