Lab 1

Student ID:

1A Problem Description:

A student uses her/his name in order of FIRSTNAME MIDDLENAME LASTNAME (also known as family name). We want to print name as LASTNAME FIRSTNAME MIDDLENAME. Write a JAVA program to do this required manipulation.

As students come from different parts of the world, names are printed in many different forms. Some students don't have all the 3 components; they just use FIRSTNAME followed by MIDDLENAME/SURNAME. If only two components are present, your program must print them in the same order if they use FISRTNAME and LASTNAME. The program, however, must print them in reverse order if they use FIRSTNAME and MIDDLENAME. To differentiate, you must assume that all the LASTNAMES will end in a vowel (a, e, i, o, u or A, E, I, O, U). If a student's name only has the FIRSTNAME it will be printed as it is.

Input Specification:

Name will be inputted in FIRSTNAME MIDDLENAME LASTNAME or FIRSTNAME LASTNAME or FIRSTNAME MIDDLENAME or FIRSTNAME formats. All the parts of the name will be separated by a single space. Each part of a name can contain maximum 50 characters.

Output Specification:

Name as expected based on the description above. Each part of the name must be separated by a single space.

Sample Test Case:

Test Case No	Test Case Input	Test Case Output
1	Prithviraj Dajisaheb Chavan	Chavan Prithviraj Dajisaheb
2	Prithviraj D. Chavan	Chavan Prithviraj D.
3	Barack Obama	Obama Barack
4	Michael Jackson	Michael Jackson
5	Jonathan	Jonathan

Note: No extra spaces, tabs, blank lines, punctuations, or any other superfluous characters are allowed. You must use only one scanner in your code.

1B Problem Description:

Most credit card numbers are encoded with a "Check Digit". It is a digit added to a number (either at the end or the beginning) that validates the authenticity of the number.

Write a java program to validate a given credit card number. For this problem, you should make use of the LUHN Formula (Mod 10) for validating credit card numbers. The algorithm has following steps:

Step 1: Double the value of alternate digits of the credit card number beginning with the second digit from the right (the rightmost digit is the check digit.)

Step 2: Add the individual digits comprising the products obtained in Step 1 to each of the unaffected digits in the original number.

Step 3: The total obtained in Step 2 must be a number ending in zero (30, 40, 50, etc.) for the account number to be validated. The total mod 10 = 0.

For example, to validate the credit card account number 49927398716:

Step 1:	4	9	9	2	7	3	9	8	7	1	6
		x2		x2		x2		x2		x2	
	—	—	—	—	—	—	—	—	—	—	—
		18		4		6		16		2	
Step 2:	4 +(1+8) + 9 + (4) + 7 + (6) + 9 +(1+6) + 7 + (2) + 6 = 70										
Step 3:	Card number is valid because the 70/10 yields no remainder.										

Input Specification:

The input will comprise of one line of integer data which is a credit card number. The credit card number will have less than 20 digits.

Output Specification:

If the credit card number is valid, output the word “VALID”. If the credit card number is not valid, output the word “INVALID” followed by a space and the correct check digit, which is the right-most digit, which would make the credit card number valid.

Sample Test Case:

No	Sample Input	Sample Output
1	49927398716	VALID
2	513467882134	INVALID 2
3	432876126	VALID