CS 11 Exercise 06
1st Semester, AY 2018-2019

University of the Philippines, Diliman

September 25, 2018

Instructions

- For this exercise, you have to write five different computer programs.
- This exercise will serve as the problem set for the mock exam. You are encouraged to answer this exercise during the lab session, using the machines in the lab.
- Specific instructions for submission will be given by your instructor during the mock exam. The same instructions will also be used for the actual exam.
- Make sure that your algorithm works given the sample input and output. You must also check if your algorithm can also handle input other than the ones given.
- Please remove any prompt messages (e.g. `Enter number:`) when getting input. Prompt messages will mess with your output, making your solution invalid.
- See the sample input and output to guide you on what and how your program must display output.
- After the mock exam, you are still required to upload your correct solutions to UVLe, since this is still an exercise. Submission links will be available after the exam (October 1).
- Submit your solutions on or before Sunday, October 07 at 11:59pm.

1 Substitution Cipher

The substitution cipher is one of the simplest methods for encrypting text. The idea in this cipher is to substitute (or replace) each letter of the alphabet with another one. For example, suppose we have the following substitution table:

INPUT	a	b	c	d	e	f	g	h	i	j	k	l	m
OUTPUT	g	j	d	f	r	v	k	n	i	s	p	u	l

INPUT	n	o	p	q	r	s	t	u	v	w	x	y	z
OUTPUT	a	y	b	e	w	m	h	x	t	o	q	z	c

On this table, each `'a'` in our message will be replaced with 'g', 'b' will be replaced with 'j', 'c' will be replaced with 'd', and so on. Using the table, encrypting the message `"watermelon"` will result to the ciphertext `"oghrwlruya"`.

Your goal for this problem is to write a program that encrypts a message using substitution cipher.

1.1 Input

The first line of input is the number of test cases t, followed by t test cases. Each test case consists of two lines. The first line is the message to be encrypted, and the second line is the entries in the substitution table - the first letter is the substitution value for `a`, the second for `b`, the third for `c`, and so on.

1.1.1 Sample Input

```
2
watermelon shake
gjdfrvknispulaybewmhxtoqzc
you can't talk peace and have a gun
iskdvpojclqwgehbturfynmzax
```

1.2 Output

For each test case, output the resulting ciphertext. Encrypt only the lowercase alphabet symbols and leave the rest (spaces, special characters, numbers, etc.) as is.

1.2.1 Sample Output

```
oghrwlruya mngpr
ahy kie'f fiwq bvikv ied jinv i oye
```

2 Inventory Sequences

An inventory sequence is an infinite sequence of numbers where each succeeding element is generated by "reading the inventory" of the current element.

As an example, consider the following inventory sequence $S = s_0, s_1, s_2, \ldots$

$$1, 11, 21, 1211, 111221, 312211, 13112221, 1113123211, 31131112131221, \ldots$$

Here, the start of the sequence s_0 is 1. Since s_0 has one 1 $(1 - 1)$, the next element s_1 is 11. Since 11 has two 1s $(2 - 1)$, the next element s_2 is 21, which is then followed by $s_3 = 1211$, $s_4 = 111221$, and so on.

Write a program that outputs s_n of an inventory sequence S, given a starting number s_0.

2.1 Input Format

Input starts with the number of test cases t, followed by t test cases. On each test case are two numbers, separated by a space. The first number is the start of the inventory sequence, s_0, and the second number is a number n. Assume that n is nonnegative.

2.1.1 Sample Input

```
5
1 2
1 5
1 0
23 6
22 21
```

2.2 Output Format

For each test case, output s_n, or the nth number in the sequence after s_0.

2.2.1 Sample Output

```
21
312211
1
3113112221123113112221l3
22
```

3 Roman Numerals

Write a program that converts a natural number in Hindu-Arabic form to Roman Numeral form.

3.1 Input

Input will start with the number of test cases t, followed be t test cases. Each test case t is a natural number between 1 to 3999.

3.1.1 Sample Input

```
10
1009
43
672
928
9999
2948
24
589
392
4290
```

3.2 Output

For each test case, output the corresponding Roman Numeral Number. Output "Invalid Input" if the input is invalid.

3.2.1 Sample Output

```
MIX
XLIII
DCLXXII
CMXXVIII
Invalid Input
MMCMXLVIII
XXIV
DLXXXIX
CCCXCII
Invalid Input
```

4 Uniquely Yours

Write a program that determines all unique characters in a string.

4.1 Input

Input will start with the number of test cases t, followed be t test cases. Each test case t is a line consisting of one or more words, delimited by a space.

4.1.1 Sample Input

```
6
At nakita kita sa tagpuan ni Bathala
May kinang sa mata na di maintindihan
Tumingin kung saan sinubukan kong lumisan
At tumigil ang mundo
Nung ako'y ituro mo
Siya ang panalangin ko
```

4.2 Output

For each test case, output a line with unique characters (case insensitive) according to the number of occurrences and in cases of tie, sort them alphabetically. Ignore white space.

4.2.1 Sample Output

```
atinkbghlpsu
animdtghksy
nuaigksmblot
agimntudlo
onu'agikmrty
angiklopsy
```

5 Word Amalgamation

In millions of newspapers across the United States there is a word game called Jumble. The objective of this game is to solve a riddle, but in order to find the letters that appear in the answer it is necessary to unscramble four words. Your task is to write a program that can unscramble words.

5.1 Input

The input contains four parts:

1. A dictionary, which consists of at least one and at most 100 words, one per line.
2. A line containing XXXXXX, which signals the end of the dictionary.
3. One or more scrambled 'words' that you must unscramble, each on a line by itself.
4. Another line containing XXXXXX, which signals the end of the file.

All words, including both dictionary words and scrambled words, consist only of lowercase English letters and will be at least one and at most six characters long. (Note that the sentinel XXXXXX contains uppercase X's.) The dictionary is not necessarily in sorted order, but each word in the dictionary is unique.

5.1.1 Sample Input

```
tarp
given
score
refund
only
trap
work
earn
course
pepper
part
XXXXXX
resco
nfudre
aptr
sett
oresuc
XXXXXX
```

5.2 Output

For each scrambled word in the input, output an alphabetical list of all dictionary words that can be formed by rearranging the letters in the scrambled word. Each word in this list must appear on a line by itself. If the list is empty (because no dictionary words can be formed), output the line `"NOT A VALID WORD"` instead. In either case, output a line containing six asterisks to signal the end of the list.

5.2.1 Sample Output

```
score
******
refund
******
part
tarp
trap
******
NOT A VALID WORD
******
course
******
```