University Interscholastic League

**Computer Science Competition**

2014 District Week 1 Programming Problem Set

**Do not open this packet until instructed to begin!**

**I. General Notes**

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points. Incorrect submissions may be reworked and resubmitted, but will receive a deduction of 5 points for each incorrect submission. Deductions are only included in the team score for problems that are ultimately solved correctly.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

**II. Table of Contents**

|  |  |
| --- | --- |
| **Number** | **Name** |
| Problem 1 | Bank |
| Problem 2 | Code |
| Problem 3 | DNA |
| Problem 4 | Exponentiation |
| Problem 5 | Fillings |
| Problem 6 | JSON |
| Problem 7 | M&M's |
| Problem 8 | Rabbits |
| Problem 9 | Right Hand |
| Problem 10 | Stealing Gold |
| Problem 11 | Tale of a Tail |
| Problem 12 | Teams |

****

**2. Code**

**Program Name: Code.java Input File: code.dat**

In light of the recent NSA revelations regarding cyber security, you have decided to encode all of your Facebook messages to a friend. You first devise a simple replacement cypher of mapping every letter to a different letter, sending your friend a shuffled string of the 26 characters a - z, with ‘a’ mapping to the first letter in the shuffled string, ‘b’ to the second letter, and so on.

For example, if the shuffled string was “zikhmpxjfsvqdaolywtecrbugn”, and the text to encode was “test”, the newly encoded string would be “emte”, since “t” maps to “e” in the shuffled string, “e” maps to “m”, and “s” maps to “t”, as you can see in the diagram below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| z | i | k | h | m | p | x | j | f | s | v | q | d | a | o | l | y | w | t | e | c | r | b | u | g | n |

To strengthen the encoding process even more, you add an offset value to the process. For example, with an offset value of 1, the encoded letter “a” becomes “b”, “b” becomes “c”,…, “y” becomes “z”, and “z” becomes “a”. Therefore the encoded string, “emte” becomes “fnuf”, the final encoded string for “test”.

In a decoding example with “dgr”, subtract 1 from each character to get “cfq”, and in the above mapping, “u” maps to “c”, “i” maps to “f”, and “l” maps to “q”, which results in “uil” as the decoded message.

Your job is to write a program that both sends (encodes) and receives (decodes) messages according to the process described above. If a character in a string is not a lowercase letter, leave it as is.

**Input**

The first line will be a single integer N, the number of protocols to process. The first line for each protocol is the shuffled string of 26 characters, and the integer offset. The bounds on this offset are -26 < offset < 26. The next line is a single integer M, the number of messages to encrypt or decrypt. Each message is on a single line, with ‘E’ or D’ for encode or decode, followed by the message to process.

**Output**

For each protocol, output “Protocol N” on its own line, followed by the resulting string from each message, each on its own line, with a blank line after each protocol set.

**Example Input File**

2

zikhmpxjfsvqdaolywtecrbugn 1

2

E UIL Computer Science rocks

D dgr 2014 xplwu

xyikbnmpvaqouctdjszerwlhfg -2

3

E sup bro!

E abcdefg

D 3947

**Example Output to Screen**

Protocol 1

UIL Cpemdfnx Slgnbln xplwu

uil 2014 rocks

Protocol 2

xpb wqr!

vwgizlk

3947

















