University of Computer Studies, Yangon

University Programming Contest

June 1, 2016

Problem Set

Please check that you have **5** problems and **11** sheets (excluding additional materials).

Problem A: Counting Weekends of the Month

Run Time Limit: 1 sec

There are seven days in a week namely Sunday (SUN), Monday (MON), Tuesday (TUE), Wednesday (WED), Thursday (THU), Friday (FRI), and Saturday (SAT). There are twelve months in a year, January (JAN), February (FEB), March (MAR), April (APR), May (MAY), June (JUN), July (JUL), August (AUG), September (SEP), October (OCT), November (NOV), and December (DEC). These months have 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30 and 31. In leap years, the month of February has 29 days. In the bracket the three letter code for each month and each day are shown. Given a month and the name of the first day of that month, you will have to find out the total number of weekend days in that month.

Input

The first line contains integer T(1<=T<=20) which is the number of test cases. The input for each test case is given in a single line. This line contains 2 strings **MTH** and **DAY**: here **MTH** is the three digit code of the month and a **DAY** is the three digit code for the name of the first day of that Month.

Output

For each line of input, produce one line of output. It contains a single integer which denotes the number of weekend days (Saturday and Sunday) in that month. You must do your calculation assuming that the year is 2016. See the samples for the exact format of output.

Sample Input	Sample Output
3 JAN FRI AUG MON DEC THU	10 8 9

This page is intentionally left blank.

Problem B. Secured Code Pattern

Run Time Limit: 1 sec

There was a software company which emphasizes working on multiple security products. The management has requested one of the new comers who recently joined the company to implement some form of communication channel to send and receive the numbers which are used in very highly secured trade transactions.

The new comer has proposed a data sending entity which follows a specific code pattern as mentioned in the table below:

Character	Coded as
0	10
1	011
2	110
3	0010
4	1110
5	1111
6	00000
7	00001

As a senior developer in the company, you need to verify the proposal from the new comer whether the code pattern can be used in the secured trade transactions. And you also need to write a simple Java program to test the pattern. The receiving entity received the following sequence of coded data, 000010000011111010011, what could be the sequence of characters received?

Input

The first line contains integer T(1<=T<=20) which is the number of test cases. The input for each test case is given in a single line. Each line contains a sequence of coded data.

Output

For each line of input, produce a line of output which is the sequence of characters.

Sample Input	Sample Output
3	
111110001000001100010	Case 1: 503703
001011100111111000000001	Case 2: 341567
000011010000010000001110	Case 3: 700764

Problem C. IMEI Validation

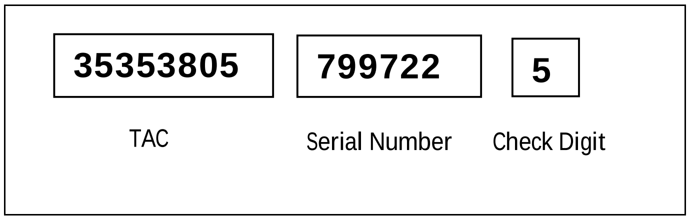
Run Time Limit: 1 sec

The International Mobile Equipment Identity (IMEI) is a unique 15 digit number used to identify mobile phones as well as some of the satellite phones. Every mobile phone, GSM modem or device with a built-in phone / mode has a unique 15 digital IMEI number. This IMEI number is used by GSM network to identify valid devices and therefore can be used to stop a stolen phone from accessing the network.

For example, if a mobile phone is stolen, the owner can call the network provider and instruct them to block the phone using its IMEI number. This renders the phone useless, regardless of whether the phone's SIM is changed.

To find the IMEI number on your mobile device, simply type *#06# and a 15 digit number will be displayed.

The structure of the IMEI consist the first eight digits, known as the Type Allocator Code (TAC) and the next six digits are the serial number specified by the equipment manufacturer. The last digit is the check digit.



The check digit can be used to validate the TAC and serial number. The check digit is calculated using the Luhn's algorithm and the steps to calculate the check digit are as follows:

Step 1: Starting from the right (excluding the check digit), double a digit every two digits.

Step 2: Sum all the digits. If the result of step 1 is a two digit number, add each individual digit together. (eg: 16 becomes 7)

Step 3: Check if the sum is divisible by 10. If the remainder is zero then that is the check digit.

Step 4: If the remainder is not zero, then subtract the remainder from 10. The resulting number will be the check digit.

Please validate the check digit of any IMEI number.

Input

The first line is the number of cases to be tested which contains integer T(1<=T<=20). The input for each test case is given in a single line. It is the IMEI number to be used in the validation.

Output

Prompt 1 or 0 whether the IMEI number input is valid or not.

Sample Input	Sample Output
3 353538057997225 352939072105730 353051013170287	Case 1:35353805 799722 5 1 Case 2:35293907 210573 0 1 Case 3:35305101 317028 7 0

Problem D. Language Recognition

Run Time Limit: 1 sec

Consider the language of sheep, which consists of strings that look like the following:

baa!, baaa!, baaaa!, baaaaa!, baaaaaa!, baaaaaa! … .

This language consists of strings with a *b*, followed by at least 2 *a*’s, followed by an exclamation point. The task is to implement a sheep language recognizer. Your recognizer should print ‘accept’ if the input string is in sheep language otherwise print ‘reject’.

Input

First line of the input will be a positive integer T(<= 10), number of test cases. Test cases are entered line by line. Each case consists of an input test string. Test string can be any strings such as baa!, baaaaaaaa!, abbbbb!, abccc, abcde, a122, etc.

Output

For each of the test, print the case number and the answer. The input string is tested and if it is in sheep language, output is ‘Accept’ and the string. If it is not in sheep language, output is ‘Reject’.

Sample Input	Sample Output
5	Case 1 : Accept, baa!
baa!	Case 2 : Reject
ba!	Case 3: Accept, baaaaaaaaaaaaaaaaa!
baaaaaaaaaaaaaaaaa!	Case 4: Reject
abbbbbb	Case 5: Reject
boooo!	

This page is intentionally left blank.

Problem E. Secret Message

Run Time Limit: 1 sec

Encryption is the conversion of electronic data into another form, called cipher text, which cannot be easily understood by anyone except authorized parties. The primary purpose of encryption is to protect the confidentiality of digital data stored on computer systems or transmitted via the Internet or other computer networks.

Alice wants to send messages to Bob. She has decided to encrypt her message so that nobody else could understand them. The code is based on shifting approach where each letter is shifted a certain number of places left or right through the alphabet. In this context, the alphabet is treated as being circular so that the first letter follows after the last letter, and the last letter precedes the first letter.

Alice applies these ideas separately to uppercase letters, lower case letters, and digits. For example, with a shift of 1, 'A' becomes 'B', 'Z' becomes 'A', 'a' becomes 'b', 'z' becomes 'a', '0' becomes '1', '9' becomes '0'. Spaces, punctuation, and any other symbols are not affected in this scheme.

Your task is to help Alice encrypt her messages.

Input

Each line of input begins with a number representing the shift. The number will be in the range -1,000,000,000 to 1,000,000,000. The number is followed by a colon (':'). The rest of the line consists of a string of 1 to 200 arbitrary characters and represents a fragment of the text to be encrypted. A single '#' on a line by itself indicates the end of input.

Output

Output will be the corresponding encrypted text fragments, one per line.

Sample Input	Sample Output
0:Plain Text	Plain Text
1:Hello Bob	Ifmmp Cpc
-1:How are you?	Gnv zqd xnt?
2:My contact number is 09970693885	Oa eqpvcev pwodgt ku 21192815007
#	