2019 ICPC Texas State – Mexico Invitational Programming Contest

Apr. 13, 2019

Problem 1: Perfect Numbers

Description

If *a*, *b*, *c* are integers such that a = bc, a is called a multiple of *b* or of *c*, and *b* or *c* is called a divisor or factor of *a*. If *c* is not ± 1, *b* is called a proper divisor of *a*. A perfect number is a positive integer that is equal to the sum of all its positive, proper divisors; for example, 6, which equals 1 + 2 + 3, and 28, which equals 1 + 2 + 4 + 7 + 14, are perfect numbers. A positive number that is not perfect is either deficient or abundant according to whether the sum of its positive, proper divisors is smaller or larger than the number itself. Thus, 9, with proper divisors 1, 3, is deficient; 12, with proper divisors 1, 2, 3, 4, 6, is abundant." Given a number, determine if it is perfect, abundant, or deficient.

Input Specification

A list of N positive integers (none greater than 60,000), with 1 < N < 100. A 0 marks the end of the list.

Output Specification

The first line of output should read PERFECTION OUTPUT. The next N lines of output should list for each input integer whether it is perfect, deficient, or abundant, as shown in the example below. Format counts: the echoed integers should be right justified within the first 5 spaces of the output line, followed by two blank spaces, followed by the description of the integer. The final line of output should read END OF OUTPUT.

Sample Input and Output

Sample Input	Sample Output
15 28 6 56 60000 22 496 0	PERFECTION OUTPUT 15 DEFICIENT 28 PERFECT 6 PERFECT 56 ABUNDANT 60000 ABUNDANT 22 DEFICIENT 496 PERFECT END OF OUTPUT

Problem 2: Goldbach's Conjecture

In 1742, Christian Goldbach, a German amateur mathematician, sent a letter to Leonhard Euler in which he made the following conjecture:

Every even number greater than 4 can be written as the sum of two odd prime numbers. For example: 8=3+5. Both 3 and 5 are odd prime numbers. 20=3+17=7+13; 42=5+37 =11+31=13+29=19+23.

Your task is to verify Goldbach's conjecture for a set of even numbers that are less than a million.

Input Specification

The input file will contain one or more even integers. Each even integer is greater than 6 and less than 1,000,000. A 0 marks the end of the input.

Output Specification

For each even integer, print one line of the form $n = a + b$, where a and b are odd primes. Numbers and operators should be separated by exactly one blank space like in the sample output below. If there is more than one pair of odd primes adding up to n, choose the pair where the difference $b - a$ is maximized. If there is no such pair, print a line saying "Goldbach's conjecture is wrong."

Sample Input and Output

Sample Input	Sample Output
8 20 42 0	8 = 3 + 5 20 = 3 + 17 42 = 5 + 37

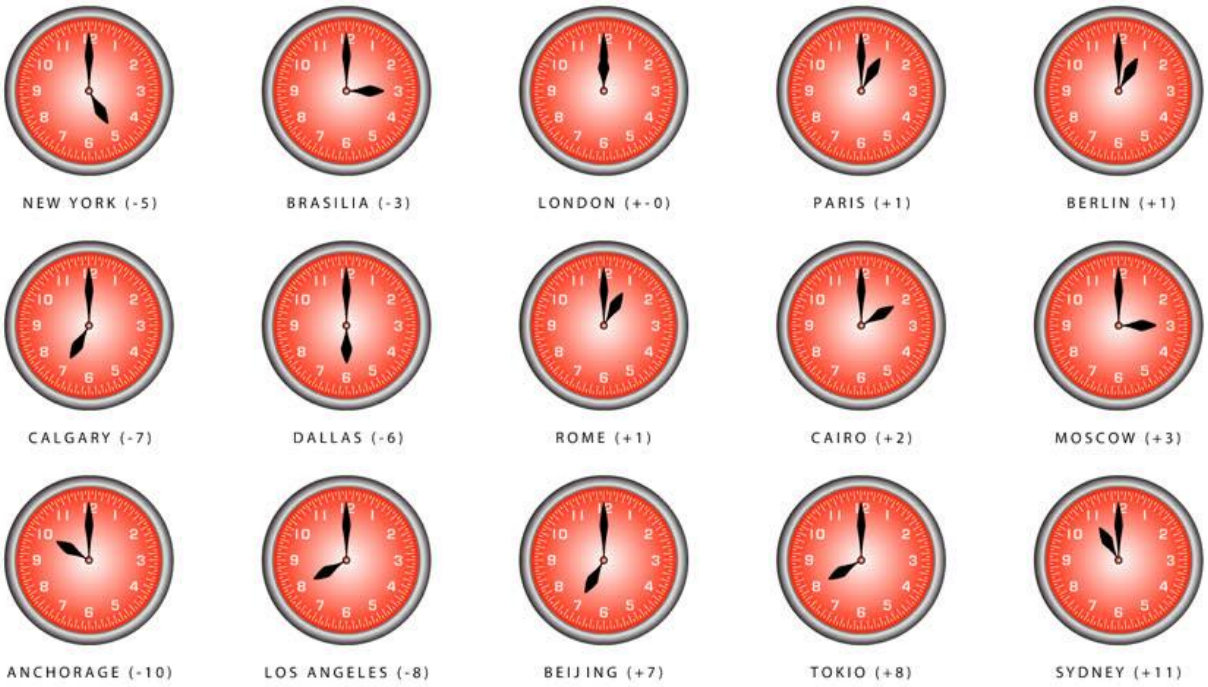
Problem 3: Time Zones

Description

Prior to the late nineteenth century, time keeping was a purely local phenomenon. Each town would set their clocks to noon when the sun reached its zenith each day. A clockmaker or town clock would be the "official" time and the citizens would set their pocket watches and clocks to the time of the town - enterprising citizens would offer their services as mobile clock setters, carrying a watch with the accurate time to adjust the clocks in customer's homes on a weekly basis. Travel between cities meant having to change one's pocket watch upon arrival. However, once railroads began to operate and move people rapidly across great distances, time became much more critical. In the early years of the railroads, the schedules were very confusing because each stop was based on a different local time. The standardization

of time was essential to efficient operation of railroads. In 1878, Canadian Sir Sanford Fleming proposed the system of worldwide time zones that we use today. He recommended that the world be divided into twenty-four time zones, each spaced 15 degrees of longitude apart. Since the earth rotates once every 24 hours and there are 360 degrees of longitude, each hour the earth rotates one-twenty-fourth of a circle or 15° of longitude. Sir Fleming's time zones were heralded as a brilliant solution to a chaotic problem worldwide.

United States railroad companies began utilizing Fleming's standard time zones on November 18, 1883. In 1884 an International Prime Meridian Conference was held in Washington D.C. to standardize time and select the Prime Meridian. The conference selected the longitude of Greenwich, England as zero degrees longitude and established the 24 time zones based on the Prime Meridian. Although the time zones had been established, not all countries switched immediately. Though most U.S. states began to adhere to the Pacific, Mountain, Central, and Eastern time zones by 1895, Congress didn't make the use of these time zones mandatory until the Standard Time Act of 1918.



Today, many countries operate on variations of the time zones proposed by Sir Fleming. All of China (which should span five time zones) uses a single time zone - eight hours ahead of Coordinated Universal Time (known by the abbreviation UTC - based on the time zone running through Greenwich at 0° longitude). Russia adheres to its designated time zones although the entire country is on permanent Daylight Saving Time and is an hour ahead of their actual zones. Australia uses three time zones - its central time zone is a half-hour ahead of its designated time zone. Several countries in the Middle East and South Asia also utilize half-hour time zones. Since time zones are based on segments of longitude and lines of longitude narrow at the poles, scientists working at the North and South Poles simply use UTC time. Otherwise, Antarctica would be divided into 24 very thin time zones! Time zones have recently been given standard capital-letter abbreviations as follows:

UTC Coordinated Universal Time:

GMT Greenwich Mean Time, defined as UTC
BST British Summer Time, defined as UTC+1 hour
IST Irish Summer Time, defined as UTC+1 hour
WET Western Europe Time, defined as UTC
WEST Western Europe Summer Time, defined as UTC+1 hour
CET Central Europe Time, defined as UTC+1
CEST Central Europe Summer Time, defined as UTC+2
EET Eastern Europe Time, defined as UTC+2
EEST Eastern Europe Summer Time, defined as UTC+3
MSK Moscow Time, defined as UTC+3
MSD Moscow Summer Time, defined as UTC+4
AST Atlantic Standard Time, defined as UTC-4 hours
ADT Atlantic Daylight Time, defined as UTC-3 hours
NST Newfoundland Standard Time, defined as UTC-3.5 hours
NDT Newfoundland Daylight Time, defined as UTC-2.5 hours
EST Eastern Standard Time, defined as UTC-5 hours
EDT Eastern Daylight Saving Time, defined as UTC-4 hours
CST Central Standard Time, defined as UTC-6 hours
CDT Central Daylight Saving Time, defined as UTC-5 hours
MST Mountain Standard Time, defined as UTC-7 hours
MDT Mountain Daylight Saving Time, defined as UTC-6 hours
PST Pacific Standard Time, defined as UTC-8 hours
PDT Pacific Daylight Saving Time, defined as UTC-7 hours
HST Hawaiian Standard Time, defined as UTC-10 hours
AKST Alaska Standard Time, defined as UTC-9 hours
AKDT Alaska Standard Daylight Saving Time, defined as UTC-8 hours
AEST Australian Eastern Standard Time, defined as UTC+10 hours
AEDT Australian Eastern Daylight Time, defined as UTC+11 hours
ACST Australian Central Standard Time, defined as UTC+9.5 hours
ACDT Australian Central Daylight Time, defined as UTC+10.5 hours
AWST Australian Western Standard Time, defined as UTC+8 hours

Given the current time in one time zone, your task is to write a program that computes what time it is in another time zone.

Input Specification

The first line of input contains N, the number of test cases. For each case a line is given with a time, and 2 time zone abbreviations. Time is given in standard a.m./p.m. format with midnight denoted "midnight" and noon denoted "noon".

Output Specification

For each case, given the current time in the first time zone, calculates the time in the second time zone.

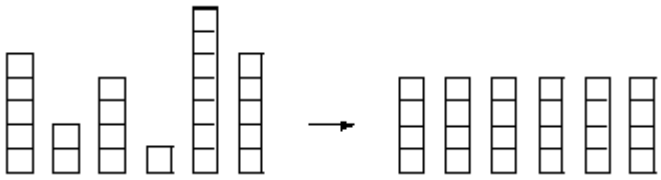
Sample Input and Output

Sample Input	Sample Output
4 noon HST CEST 11:29 a.m. EST GMT 6:01 p.m. CST UTC 12:40 p.m. ADT MSK	midnight 4:29 p.m. 12:01 a.m. 6:40 p.m.

Problem 4: Box of Bricks

Description

Little Bob likes playing with his box of bricks. He puts the bricks one upon another and builds stacks of different height. "Look, I've built a wall!", he tells his older sister Alice. "Nah, you should make all stacks the same height. Then you would have a real wall.", she retorts. After a little consideration, Bob sees that she is right. So he sets out to rearrange the bricks, one by one, such that all stacks are the same height afterwards. But since Bob is lazy he wants to do this with the minimum number of bricks moved. Can you help?



Input Specification

The input consists of several data sets. Each set begins with a line containing the number n of stacks Bob has built. The next line contains n numbers, which are the heights h_i of the n stacks. You may assume $1 \leq n \leq 50$ and $1 \leq h_i \leq 100$. The total number of bricks will be divisible by the number of stacks. Thus, it is always possible to rearrange the bricks such that all stacks have the same height. The input is terminated by a set starting with $n = 0$, which should not be processed.

Output Specification

For each set, first print the number of the set, as shown in the sample output. Then print the line "The minimum number of moves is k.", where k is the minimum number of bricks that have to be moved in order to make all the stacks the same height. Output a blank line after each set.

Sample Input and Output

Sample Input	Sample Output
6 5 2 4 1 7 5 0	Set #1 The minimum number of moves is 5.

Problem 5: Elven Postman

Description

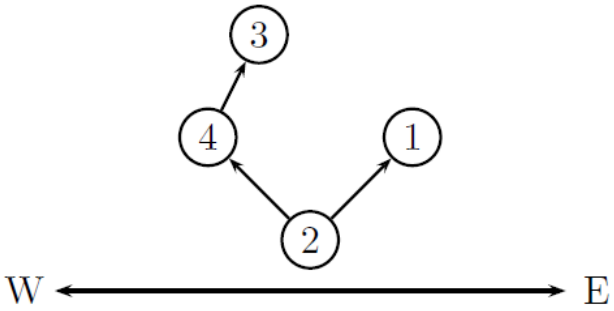
Elves are very peculiar creatures. As we all know, they can live for a very long time and their magical prowess are not something to be taken lightly. Also, they live on trees. However, there is something about them you may not know. Although delivering stuffs through magical teleportation is ex-tremely convenient (much like emails). They still sometimes prefer other more “traditional” methods.

So, as a elven postman, it is crucial to understand how to deliver the mail to the correct room of the tree. The elven tree always branches into no more than two paths upon intersection, either in the east direction or the west. It coincidentally looks awfully like a binary tree we human computer scientist know. Not only that, when numbering the rooms, they always number the room number from the east-most position to the west. For rooms in the east are usually more preferable and more expensive due to they having the privilege to see the sunrise, which matters a lot in elven culture.

Anyways, the elves usually wrote down all the rooms in a sequence at the root of the tree so that the postman may know how to deliver the mail. The sequence is written as follows, it will go straight to visit the east-most room and write down every room it encountered along the way. After the first room is reached, it will then go to the next unvisited east-most room, writing down every unvisited room on the way as well until all rooms are visited.

Your task is to determine how to reach a certain room given the sequence written on the root.

For instance, the sequence 2, 1, 4, 3 would be written on the root of the following tree.



Input Specification

First you are given an integer T ($T \leq 10$) indicating the number of test cases.

For each test case, there is a number n ($n \leq 1000$) on a line representing the number of rooms in this tree. n integers representing the sequence written at the root follow, respectively $a1, \ldots, an$ where $a1, \ldots, an \in \{1, \ldots, n\}$.

On the next line, there is a number q representing the number of mails to be sent. After that, there will be q integers $x1, \ldots, xq$ indicating the destination room number of each mail.

Output Specification

For each query, output a sequence of move (E or W) the postman needs to make to deliver the mail. For that E means that the postman should move up the eastern branch and W the western one. If the destination is on the root, just output a blank line would suffice.

Note that for simplicity, we assume the postman always starts from the root regardless of the room he had just visited.

Sample Input and Output

Sample Input	Sample Output
2 4 2 1 4 3 3 1 2 3 6 6 5 4 3 2 1 1 1	E WE EEEEE

Problem 6: Birthday Gift

Description

Princess Diana invites her friends to come to her birthday party. Each of her friends will bring a gift of some value *v*, and all of them will come at a different time. Because the lobby is not large enough, Diana can only let a few people in at a time. She decides to let the person whose gift has the highest value enter first. Each time when Diana opens the door, she can decide to let *p* people enter her castle. If there are less then *p* people in the lobby, then all of them would enter. And after all of her friends have arrived, Diana will open the door again and this time every friend who has not entered yet would enter. If there are two friends who bring gifts of the same value, then the one who comes first should enter first. Given a query *n*, please tell Diana who the *n-th* person to enter her castle is.

Input Specification

The 1st line of the input gives the number of test cases, T, where $1 \leq T \leq 15$.

In each test case, the 1st line contains three numbers *k*, *m* and *q* separated by blanks. *k* ($1 \leq k \leq 150{,}000$) is the number of friends being invited. The door will open *m* ($0 \leq m \leq k$) times before all Diana's friends arrive. Diana will have q ($1 \leq q \leq 100$) queries. The *i-th* of the following *k* lines gives a string B_i, which consists of up to 200 characters, and an integer v_i, *where* $1 \leq v_i \leq 10^8$, separated by a space. B_i is the name of the *i-th* person coming to Diana's party and B_i brings a gift of value v_i.

Each of the following *m* lines contains two integers t ($1 \leq t \leq k$) and p ($0 \leq p \leq k$), which are separated by a space. The door will open right after the *t-th* person arrives, and Diana will let *p* friends enter her castle. The last line of each test case contains *q* numbers $n_1, \ldots, n_q$ (separated by a space), which indicates Diana wants to know who are the n_1*-th*, . . . , n_q*-th* friends to enter her castle.

Output Specification

For each test case, output the corresponding name of Diana's query, separated by a space.

Sample Input	Sample Output
1 5 2 3 Elena 3 Alice 3 Maltran 3 Elizabeth 5 Mikleo 6 1 1 4 2 1 2 3	Elena Elizabeth Alice