


UCLA-ve Local Programming Contest 2011
Contest Session
Universidad Centroccidental "Lisandro Alvarado"

Sponsored by:

July 09, 2011

This problem set should contain a rules section and eight (08) problems on nine (09) numbered pages.
Please inform the Contest Staff immediately if something is missing from your problem set.
Welcome to the Programming Contest and enjoy it!.
http://uclave.acm.org/

Rules

- Each team will be provided with a single computer that have the VM of ACM-ICPC running on VirtualBox. All teams have equivalent computing equipment.
- There are eight (08) problems for each team to be completed in five (05) hours.
- All problems require that you read test data from the *standard input* and write results to the *standard output.*
- You have may use any of the following programming languages: C, C++ or JAVA. You may use different programming languages for different problems and even for different submissions of a problem.
- All problems have a source file named with the first word of title of the problem.. The extension you must use for your source files are: *.c* for programs written in C, *.cpp or .cc* for those in C++, and *.java* for those in JAVA.
- You may use any of the standard libraries that your chosen programming languages provides. You may not use any other library that requires an extra flag to be passed to the compiler command. If you do this, judges will probably get compilation-linking error in your program.
- Output correspond exactly to the provided sample output format, including (mis)spelling and spacing. Multiple spaces will not be used in any of the judge's output, except where explicitly stated.
- Your solution to any problem should be submitted for judging using PC^2 software only. Once you have submitted a solution, it will reach the judges. The time it takes for your problem to be judged will depend on how busy the judges are. Once your submission has been judged, you will receive a message through PC^2 indicating the judgment. If your solution is accepted the message will be "Yes", if it is not then the message will be "No", along with the type of error the judges encountered, which may be: "Compilation Error", "Runtime Error", "Time-Limit Exceeded", "Wrong Answer" or "Presentation Error".
- Programming style is not considered in this contest. The judges will only test whether the *input / output* behavior of your program is correct or not. However, each problem has an execution time-limit of 60 seconds. That is, if your program takes more that 60 seconds to execute for the given input, it will be judged as incorrect.
- Contestants may bring any printed materials *(books, papers, documentation, source code of programs, etc.)* to the contest area, but no soft copy will be allowed *(diskettes, CD's, DVD's, pendrives, etc).*

Problem A. Student Chapter

Input file: `Standard Input`
Output file: `Standard Output`

Recently, your team that is a member of ACM student chapter has noticed that the computer that use to practice for programming contests is not good enough anymore. Therefore, you purposed with the student chapter chair to buy a new computer. To make the ideal computer for your needs, you decide with the chair buy separate components and reinforce on the computer. You need to buy exactly one of each type of component. The problem is which components to buy. As you all know, the quality of a computer is equal to the quality of its weakest component. Therefore, you want to maximize the quality of the component with the lowest quality, while not exceeding budget of ACM student chapter.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with two integers: $1 \leq n \leq 1000$, the number of available components and $1 \leq b \leq 1000000000$, budget of ACM student chapter.
- n lines in the following format: **"type name price quality"**, where type is a string with the type of the component, name is a string with the unique name of the component, price is an integer ($0 \leq price < 1000000$) which represents the price of the component and quality is an integer ($0 \leq quality \leq 1000000000$) which represents the quality of the component (higher is better). The strings contain only letters, digits and underscores and have a maximal length of 20 characters.

It will always possible to construct a computer with the budget of the ACM student chapter.

Output

Per testcase:

- One line with one integer: the maximal possible quality.

Sample input and output

Standard Input	Standard Output
1 13 800 processor 3500MHz 66 5 processor 6000MHz 219 12 memory 2GB 35 3 memory 8GB 170 12 mainbord all-onboard 52 10 harddisk 500GB 54 10 harddisk 1TB 99 12 casing thermaltake 36 10 monitor 20inch 210 9 monitor 22inch 293 12 mouse wireless-optical 18 12 mouse apple 30 9 keyboard logitech 4 10	9

Problem B. WikiSpy F1

Input file: `Standard Input`
Output file: `Standard Output`

WikiSpy F1 is the most recent form of industrial spying made for F1 teams. Joe is an industrial spy – don't tell anybody! His most recent job was to steal the latest inventions from McLaren aerodynamic lab. It was hard to obtain some of their results but He got their waste out of a document shredder. He comment us that have already reconstructed the research topic is fast factorization. But the remaining paper snippets only have single digits on it and he cannot imagine what they are for. Could it be that those digits form prime numbers? Please help to find out how many prime numbers can be formed using the given digits.

Input

The first line of the input holds the number of test cases $T(1 \leq T \leq 200)$. Each test case consists of a single line. This line contains the digits (at least one, at most seven) that are on the paper snippets.

Output

For each test case, print one line containing the number of different primes that can be reconstructed by shuffling the digits. You may ignore digits while reconstructing the primes (e.g., if you get the digits 7 and 1, you can reconstruct three primes $7, 17$, and 71). Reconstructed numbers that (regarded as strings) differ just by leading zeros, are considered identical (see the fourth case of the sample input).

Sample input and output

Standard Input	Standard Output
4 1276543 17 9999999 011	1336 3 0 2

Problem C. Smart-G

Input file: `Standard Input`
Output file: `Standard Output`

Smart-G is a revolutionary invention of hardware and software created by CODECERC in the parking lot business: Smart Garage. The concept is very simple: you drive your car into the elevator at the entrance of the tower where is installed Smart-G, and the elevator and conveyor belts drag the car to an empty parking spot, where the car remains until you pick it up. When you return, the elevator and conveyor belts move your car back to the entrance and you're done. The tower with Smart-G her design is very simple. There is one central elevator that transports the cars between the different floors. On each floor there is one giant circular conveyor belt on which the cars stand. This belt can move in clockwise and counterclockwise direction. When the elevator arrives on a floor, it becomes part of the belt so that cars can move through it. At the end of the day the tower with Smart-G is usually packed with cars and a lot of people come to pick them up, that is a really challenge for this product. Customers are processed in a first come first serve order: the elevator is moved to the floor of the first car, the conveyor belt moves the car on the elevator, the elevator is moved down again, and so on. We like to know how long it takes before the last customer gets his car. Moving the elevator one floor up- or downwards takes 10 seconds and moving a conveyor belt one car in either direction takes 5 seconds.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with two integers H and L with $1 \leq H \leq 50$ and $2 \leq L \leq 50$: the height of the tower with Smart-G and the length of the conveyor belts.
- H lines with L integers: the initial placement of the cars. The jth number on the ith line describes the jth position on the ith floor. This number is -1 if the position is empty, and r if the position is occupied by the rth car to pick up. The positive numbers form a consecutive sequence from 1 to the number of cars. The entrance is on the first floor and the elevator (which is initially empty) is in the first position. There is at least one car in the parking tower.

Output

Per testcase:

- One line with the number of seconds before the last customer is served.

Sample input and output

Standard Input	Standard Output
2 3 6 -1 5 6 -1 -1 3 -1 -1 7 -1 2 9 -1 10 4 1 8 -1 1 5 -1 2 1 -1 3	320 25

Problem D. Orinoco Rafting

Input file: `Standard Input`
Output file: `Standard Output`

You have been hired by a theme park in Venezuela to design a new attraction in the Orinoco river: a water rafting ride. You already designed the track; it is a round trip that is described by an inner and an outer polygon. The space in between the two polygons is the track. You still need to design the rafts, however. It has been decided that they should be circular, so that they can spin freely along the river track and increase the fun and excitement of the ride. Besides that, they should be as big as possible to fit the maximum number of people, but they can't be too big, for otherwise they would get stuck somewhere on the river track. What is the maximum radius of the rafts so that they can complete the river track?

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with an integer $I (3 \leq I \leq 100)$: the number of points of the inner polygon.
- I lines with two integers each: the coordinates of the points of the inner polygon in consecutive order.
- One line with an integer $O (3 \leq O \leq 100)$: the number of points of the outer polygon.
- O lines with two integers each: the coordinates of the points of the outer polygon in consecutive order.

All coordinates have absolute value no larger than 1000. The points of the polygons can be given in either clockwise or counterclockwise order and the two polygons do not intersect or touch themselves or each other. The outer polygon encloses the inner polygon.

Output

Per testcase:

- One line with a floating point number: the maximal radius of the white water rafts. This number must have a relative or absolute error less than 10^{-6}.

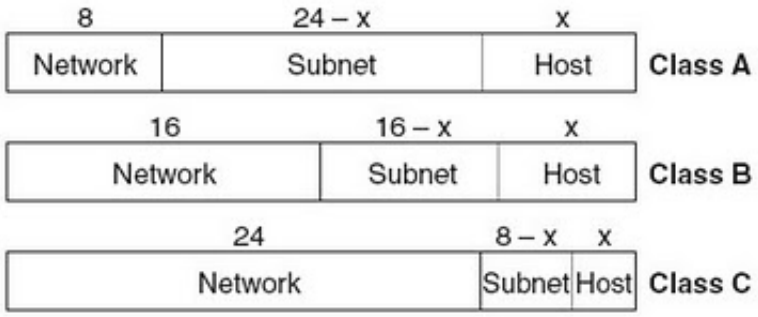
Sample input and output

Standard Input	Standard Output
1 3 0 0 1 0 1 1 5 3 -3 3 3 -4 2 -1 -1 -2 -2	0.70710678

Problem E. IPv4 Again

Input file: `Standard Input`
Output file: `Standard Output`

A subnetwork, or subnet, is a logically visible subdivision of an IP network. The practice of dividing a network into sub-networks is called subnetting. All computers that belong to a subnet are addressed with a common, identical, most-significant bit-group in their IP address. This results in the logical division of an IP address into two fields, a network or routing prefix and the rest field. The rest field is a specific identifier for the computer or the network interface.



Carl's the network administrator of UASC is on leave and we need to subnet our IP networks so that there are M additional networks. Your job is to write a program that provides a correct subnet mask for the given IP Address.

Input

Each input line contains an IP Address and M the number of additional networks.

Output

For each corresponding IP and M Output the subnet mask in a new line.

Sample input and output

Standard Input	Standard Output
192.168.24.0 2	255.255.255.128
130.186.0.0 11	255.255.240.0
200.44.32.0 5	255.255.255.224

Problem F. Geek's birthday

Input file: `Standard Input`
Output file: `Standard Output`

It's your best friend's birthday, and you and some other friends decided to buy him a Sims3, because who wouldn't want to have that? You agreed to divide the costs as fairly as possible. Since some of you have more money available than others, you also agreed that nobody has to pay more than he can afford. Every contribution will be a multiple of 1 Bolivar, i.e., nobody can pay fractions of a Bolivar. Everybody tell down the maximum amount he is able to contribute. Taking into account these maximum amounts from everybody, you share the cost of the present as fairly as possible. That means, you minimize the largest distance of the contributions to $\frac{1}{n}$-th of the total cost. In case of a tie, minimize the second largest distance, and so on. Since the smallest unit of contribution is 1 Bolivar, there might be more than one possible division of the cost. In that case, persons with a higher maximum amount pay more. If there is still ambiguity, those who come first in the list pay more. Since you bought the present, it is your task to figure out how much every friend has to pay (including you).

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- One line with two integers P and N: the price of the present in bolivars ($1 \leq P \leq 1000000$) and the number of people ($2 \leq N \leq 100$) who contribute to the present (including you).
- One line with N integers $A_i (1 \leq A_i \leq 1000000)$, where ai is the maximum amount, in bolivars, that the i-th person on the list is able to contribute.

Output

Per test case:

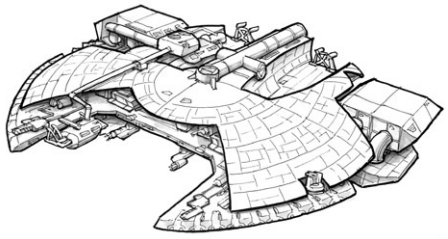
- One line with N integers: the amounts each person has to contribute according to the scheme. If the total cost cannot be divided according to the above rules, the line must contain **IMPOSSIBLE** instead.

Sample input and output

Standard Input	Standard Output
3 20 4 10 10 4 4 34 5 9 8 9 9 4 7 3 1 1 4	6 6 4 4 8 7 8 7 4 IMPOSSIBLE

Problem G. Blackholes

Input file: Standard Input
Output file: Standard Output



You and your Jedi master, have built a new spaceship and want to explore space with it. During his first travel, your jedi master discovered that the universe is full of blackholes. These blackholes allow one to travel to places far, far away, but moreover, they can also send you to times long ago or in the distant future. Having mapped these blackholes and their respective end points, you and your jedi master boldly decide to board his spaceship and go to some distant place you'd like to visit. Of course, you want to arrive at your destination as early as possible. The question is: what is this earliest arrival time?

Input

The first line of input contains an integer $T(1 \leq T \leq 200)$, the number of test cases. Each test case starts with a line containing two coordinate triples x_0, y_0, z_0 and x_1, y_1, z_1, the space coordinates of your departure point and destination. The next line contains an integer $N(0 \leq N \leq 50)$, the number of blackholes. Then follow N lines, one for each blackhole, with two coordinate triples x_s, y_s, z_s and x_e, y_e, z_e, the space coordinates of the blackhole entry and exit points, respectively, followed by two integers $t, d(1000000 \leq t, d \leq 1000000)$, the creation time t of the blackhole and the time shift d when traveling through the blackhole. All coordinates are integers with absolute values smaller than or equal to 10000 and no two points are the same. Note that, initially, the time is zero, and that tunneling through a blackhole happens instantly. For simplicity, the distance between two points is defined as their Euclidean distance (the square root of the sum of the squares of coordinate diferences) rounded up to the nearest integer. Your master's spaceship travels at speed 1.

Output

For each test case, print a single line containing an integer: the earliest time you can arrive at your destination.

Sample input and output

Standard Input	Standard Output
2 0 0 0 100 0 0 2 1 1 0 1 2 0 -100 -2 0 1 0 100 1 0 -150 10 0 0 0 10 0 0 1 5 0 0 -5 0 0 0 0	-89 10

Problem H. Programmer

Input file: `Standard Input`
Output file: `Standard Output`

You are programmer and working on a module that takes a piece of code containing some definitions or other tabular information and aligns each column on a fixed vertical position, while keeping the resulting code as short as possible, making sure that only whitespaces that are absolutely required stay in the code. So, that the first words on each line are printed at position $p_1 = 1$; the second words on each line are printed at the minimal possible position p_2, such that all first words end at or before position $p_2 - 2$; the third words on each line are printed at the minimal possible position p_3, such that all second words end at or before position $p_3 - 2$, etc.

For the purpose of this problem, the code consists of multiple lines. Each line consists of one or more words separated by spaces. Each word can contain uppercase and lowercase Latin letters, all ASCII punctuation marks, separators, and other non-whitespace ASCII characters (ASCII codes 33 to 126 inclusive). Whitespace consists of space characters (ASCII code 32).

Input

The input contains one or more lines of the code up to the EOF. All lines (including the last one) are terminated by a standard end-of-line. Each line contains at least one word, each word is 1 to 80 characters long (inclusive). Words are separated by one or more spaces. Lines of the code can have both leading and trailing spaces. Each line in the input is at most 180 characters long. There are at most 1000 lines in the input.

Output

Write to the standard output the reformatted, aligned code that consists of the same number of lines, with the same words in the same order, without trailing and leading spaces, separated by one or more spaces such that i-th word on each line starts at the same position p_i.

Sample input and output

The '␣' character in the example below denotes a space character (ASCII code 32).

Standard Input	Standard Output
␣␣start:␣␣integer;␣␣␣␣//␣begins␣here stop:␣integer;␣//␣␣ends␣here␣␣ ␣s:␣␣string;␣␣␣ c:␣␣␣char;␣//␣temp␣	start:␣integer;␣//␣begins␣here stop:␣␣integer;␣//␣ends␣␣␣here s:␣␣␣␣␣string; c:␣␣␣␣␣char;␣␣␣␣//␣temp