All-Ukrainian Collegiate Programming Contest
Semi-Final

April 8, 2010

Problem A

Automatic Cash Terminal

(1 sec)

Input: a.dat
Output: Standard Output

The automatic teller machine (ATM) has a lot of banknotes of two different values. When a person who uses the terminal requests some amount, the machine issues exactly this amount of money (of course, if this amount does not exceed a balance of the user's account). Most people do not like to carry a huge bundle of banknotes, therefore the machine must issue requested amount using minimal possible number of banknotes.

Your task is to write a program that determines the number of banknotes of each value that should be issued by the machine so that the user receives the requested amount and the total number of banknotes is minimal. You may assume that the number of banknotes of each value is not limited.

The first number in the input gives the number of test cases. A single line for each case contains three integers a, b (the values of banknotes in the machine) and S (the amount of money requested by the user). ($1 \le a,b \le 10000$, $a \ne b$, $0 \le S \le 10^9$). Integers in each line are separated by a space.

For each test case print a single line in the output with two integers – numbers of banknotes of each value which must be issued by ATM. In case when it's not possible to issue requested amount, print the word "Impossible" (without quotes).

input	Output
3 1 10 23 3 2 7 4 6 2	3 2 1 2 Impossible

All-Ukrainian Collegiate Programming Contest
Semi-Final

April 8, 2010

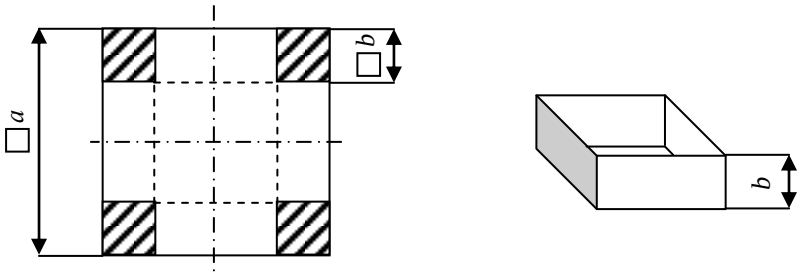
Problem B

Box

(1 sec)

Input: b.dat
Output: Standard Output

A confectionery needs to make boxes for chocolates from square cardboard sheets of size $a \times a$. This box should be opened from the top and have square base. Box is made in two steps. At the first step square pieces of size $b \times b$ get cut at all corners of the cardboard sheet. At the second step rectangular pieces get bent along the edges by the angle of 90 degrees to form the box. Having length of cardboard sheet's side a you need to find height b of the box when its volume is maximal.



The first line of input contains one integer n ($1 \le n \le 10$) - number of tests. Each of subsequent n lines contains one real number – length of cardboard sheet's side a ($1 \le a \le 1014$) for the test.

Output should contain n lines, each line should contain optimal height b for the corresponding test rounded to 10 digits after dot.

Input	Output
1 34.0	5.6666666667

All-Ukrainian Collegiate Programming Contest
Semi-Final

April 8, 2010

Problem C

Weird sort

(1 sec)

Input: c.dat
Output: Standard Output

Having a sequence of N integers a_1, a_2, …, a_N, you need to order them in a way when no two consecutive integers have consecutive values. In other words, for all i, where $0<i<N$, condition $a_i+1 \neq a_{i+1}$ should be satisfied for the final sequence.

If more than one sequence satisfying this condition exists, lexicographically minimal one should be found.

The input file consists of several data sets. In the first line of each set the sequence length N ($1 \leq N \leq 50000$) is given. The second line contains N integers a_1, a_2, …, a_N, separated by single spaces. Each integer does not exceed 10^9 in its absolute value. The value $N=0$ indicates the end of the input file.

For each data set you need to print result sequence in separate line. Integers in the sequence must be separated by single spaces. Print “No solution” (without quotes) if requested sequence does not exist.

Sample input and output

input	output
2 1 2 6 1 2 3 4 5 6 6 1 1 2 2 3 3 0	2 1 1 3 2 4 6 5 1 1 3 3 2 2

All-Ukrainian Collegiate Programming Contest
Semi-Final

April 8, 2010

Problem D

Bowling

(1 sec)

Input: d.dat
Output: Standard Output

It is well known that programmers enjoy bowling. Bowling is a competitive sport in which a player (the “bowler”) rolls a bowling ball down a wooden or synthetic lane with the objective of scoring points by knocking down as many pins as possible.

Scoring is part of bowling which beginners find hard to understand. Fortunately it is not as hard as it seems.

As most bowling centers have a scoring computer, you do not usually have to score yourself. One day, when it all breaks down, or you are needed to correct a mistake, an understanding of the scoring rules becomes important.

The most difficult part of bowling scoring is when a strike or spare is scored, as the score on the scorecard does not get updated immediately.

A game consists of ten frames, which start with a full rack of ten pins. In each frame, you have two deliveries of your ball, in which you need to knock down as many of ten pins as you can.

If you knock down all the pins on your first ball, it is called a strike. The score doesn't get added immediatelly because for a strike you get the values of your next two balls as a bonus. For example, if you score a strike in the first frame, then 7 and 1 in the second frame, you would score 18 (10+7+1) for the first frame, and 8 for the second frame, making a total of 26 after two frames.

If you knock down some of the pins on the first ball, and knocked down the remainder of the pins in the second ball, it is known as a spare. Again, the score doesn't get added immediatelly because for a spare you get the value of your next ball as a bonus. For example, if you score a spare in the first frame, say 6 and 4, then get 8 and 1 in the second frame, you would score 18 (6+4+8) for the first frame, and 9 for the second frame, making a total of 27 after two frames.

When it comes to the final frame, it is slightly different. In the final frame, you get extra balls if you strike or spare, to allow bonus points. If you strike in the first delivery you have the opportunity to strike in the remaining two and have three deliveries in total. If you scored strikes in each of your final three deliveries, the score for the final frame would be 30 (10+10+10). If you spare the final frame, you get the third delivery as a bonus. So, a spare, 9 and 1, followed by a strike would equal 20 (9+1+10).

You have to write a program which will calculate the score the player gets for the game given the information about the pins knocked down after each delivery of the ball.

The first line of the input contains number **t** <= 1000 – the amount of test cases. Then the description of each of **t** test cases follow one per line. Each test case consists of several integers 0 <= **a** <= 10 – the amount of pins knocked down after each delivery of the ball. Each test case describes a full game for one player. All the games in the input file are correct.

For each test case output the number of points the player gets in a game on a separate line.

Input	Output
3 10 10 10 10 10 10 10 10 10 10 10 10 3 5 1 5 7 1 10 1 6 10 6 2 1 2 0 5 8 1 9 1 5 0 3 0 8 1 6 4 7 2 7 1 6 3 10 4 4	300 89 101

Problem E

Exam

(2 sec)

Input: e.dat
Output: Standard Output

Many universities use scoring system where students can earn up to 100 points, out of which up to 75 — during the semester, and up to 25 — during the final exam. The final mark is determined depending on the sum of semester and examination points by the following table:

Sum of points	European mark	National mark
90—100	A	Excellent
82—89	B	Good
75—81	C	
68—74	D	Satisfactory
60—67	E	
35—59	FX	Bad

If a student gets strictly less than 35 points during the semester, they are not allowed to pass the examination; let's assume in this case that those student's names have been excluded from the list.

If you read the column of European marks in examination list from top to bottom, you can find various "words". For example, if there are consecutive sums of points 92, 75 and 66, they are marked as A, C and E respectively, and form the "word" ACE. In case of FX, both letters (first F, then X) appear in the "word".

An exam is a risky thing, it's impossible to know its results beforehand. But the lecturer knows both the approximate level of knowledge of each student and the list of exam tasks. So the lecturer can estimate the percentage probabilities for each student to earn at the exam every possible number of points: what is the probability that the student gets 0 points, 1 point, ..., 25 points – totally 26 nonnegative integers, whose sum equals 100. Points, obtained by every student during the semester, are also known (as fixed numbers from 35 to 75, without any probabilities).

The lecturer is a great esthete and dislikes situations when the "word", produced by European marks, contains any "unpleasant" substring (just as substring, i.e. when the letters are consecutive).

Your task is to write a program to find the probability that this lecturer the esthete will be satisfied because none of the "unpleasant" substrings will occur.

The first line of the input contains the number of test cases. In each test case, the first line contains the number of students N ($3 \le N \le 100$). Each of the following N lines contains 27 space-separated integers — the number of semester points (from 35 to 75), and 26 probabilities corresponding to 0, 1, 2, ..., 25 examination points (each probability is non-negative, their total sum is 100). The next line of the test case contains number K ($1 \le K \le 100$) of "unpleasant" words according to lecturer's opinion. Each of the following K test case lines contains an "unpleasant" word. It's guaranteed that each of these K lines contains only capital Roman letters (any letters, not only A-F and X). Number of letters in each line is from 2 to 15 and it ends with the end-of-line symbol.

Your program should write to output exactly one floating-point number in a single line – calculated probability (in percent) that the lecturer will be satisfied. Floating-point format may be any of the standard ones (using decimal point, not decimal comma). *The answer will be accepted if its relative error does not exceed 1e–6.*

Input	Output
1 3 72 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 2 3 5 7 9 14 16 21 12 5 1 55 0 0 0 1 2 3 4 5 6 7 8 8 9 8 8 7 6 5 4 3 2 2 1 1 0 0 55 0 0 0 1 2 3 4 5 6 7 8 8 9 8 8 7 6 5 4 3 2 2 1 1 0 0 1 DE	79.5

The 1st student's sum will be at least 72+10=82 points, so his mark can't be D. Thus, "unpleasant" word DE will occur if and only if the 2nd student gets from 13 to 19 points (the probability is 8%+8%+7%+6%+5%+4%+3%=41%) and the 3rd – from 5 to 12 (the probability 3%+4%+5%+6%+7%+8%+8%+9%=50%). So, the word DE will appear with probability 0.41*0.5=0.205, and will not appear with probability 1–0.205=0.795 (i.e. 79.5%).

All-Ukrainian Collegiate Programming Contest
Semi-Final

April 8, 2010

Problem F

Platforms

(2 sec)

Input: f.dat
Output: Standard Output

Antigravity has been widely used for a long time on the Olympia planet. The Olympian ministry of entertainment opened a new set of sightseeing platforms, that immovably hang at different heights above the Rio-ACM river. The platforms are small and may be considered as points. The river can be considered approximately straight, so all the platforms (points) are placed in a vertical plane. That is, the location of each platform can be described by two coordinates: x-coordinate specifies the horizontal distance from the river's source, y-coordinate – the height above the water. No platform is located exactly under any other one – thus, all x-coordinates are distinct.

Unfortunately, these platforms didn't prove to be popular for sightseeing, so the ministry is looking for a plan how to use them in another way – preferably such a way, which doesn't require moving of the platforms. One of the plans suggests equipping the platforms as stations for the hang gliders (also known as delta-planes). It's not interesting to use antigravity in sports, so hang gliders can fly from any platform to a platform of lower or equal height only. On religious grounds, Olympians never fly hang gliders in the direction opposite to the river flow.

According to marketing researches, the route will be popular if and only if it allows the greatest possible number of consecutive flights from one platform to another one, next one, and so on. The length of flights has no significant influence on route's popularity.

Your task is to write a program to find which of the platforms will belong to at least one of the popular routes.

The 1st line of the input contains the number of test cases T. In each test case, the first line contains the number of platforms N (1<N<123456), each of following N lines contains two non-negative integer numbers — x- and y-coordinates of the corresponding platform.

The program should write to standard output T two-lines groups — the answers for the corresponding test cases. The first line in each group should contain the number of flights in the popular routes and the total number of platforms that belong to any of the popular routes. The second line in the group should contain a list (in ascending order) of x-coordinates of these platforms. Numbers in each line should be separated by spaces.

Input	Output
2 2 2 3 4 5 5 11 4 44 5 22 2 33 3 55 1	0 0 2 4 11 22 33 55

All-Ukrainian Collegiate Programming Contest
Semi-Final

April 8, 2010

Problem G

The Very Greatest Common Divisor

(2 sec)

Input: g.dat
Output: Standard Output

You need to find greatest common divisor of two integers *a* and *b*. Each number *a* and *b* are determinants of the square matrix of the form:

$$\begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ -1 & 1 & 1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}$$

The first line of the input file contains number $n < 250$ of test cases. The description of a test case consists of two lines. The first line contains integer a ($0 < a < 10^{12540}$), the second – integer b ($0 < b < 10^{12540}$).

For each test case print the greatest common divisor of integers *a* and *b* on a separate line.

Input	Output
3 2 3 3 21 6765 610	1 3 5

All-Ukrainian Collegiate Programming Contest
Semi-Final

April 8, 2010

Problem H

Hamster 2

(1 sec)

Input: h.dat
Output: Standard Output

There is a competition of flying hamsters in Hamsterburg. Each competing hamster is thrown from a sling. The initial speed of the hamsters is $\mathbf{V_0}$ m/s. Free fall acceleration is **g** = 10 m/s^2. There is no air friction. The size of the hamster and the sling are negligible. When the hamster is thrown from the sling its altitude is 0 meters. There is a number of vertical gates in the air. Each gate has a lower and an upper bound. If we mark the points directly under each of the gates on the ground – those points are positioned in one line and on one side from the starting point. A hamster gets as many points as the amount of gates he flies through. You have to calculate the maximal amount of points that a hamster can get in one flight. It is considered that a hamster flies through the gate if he touches the bounds of the gate during the flight of flies between the bounds.

The first line of the input contains number 0 < **t** <= 10 the amount of test cases. The description of each test case follows. Each test starts with two integers 0 < $\mathbf{V_0}$ <= 1000 – the initial speed of the hamster and 0 < **n** <= 20000 – the total amount of gates. Each of the next **n** lines contains the description of one of the gates: three integers 0 < **x** <= 10000 – the distance from the starting point to the point directly under the gate, 0 < $\mathbf{y_1}$ <= $\mathbf{y_2}$ <= 10000 – lower and upper bound of the gate.

For each test case output the maximal amount of gates a hamster can fly through in one flight on a separate line.

Input	Output
3 10 2 3 1 2 3 2 3 10 3 1 1 1 2 2 3 3 4 6 10 3 1 1 2 2 3 4 3 5 6	2 1 2

Problem I

Factorials

(1 sec)

Input: i.dat
Output: Standard Output

The factorial of an integer n, written n!, is the product of all the integers from 1 through n inclusive. The factorial quickly becomes very large: 13! is too large to store in a 32-bit integer on most computers, and 70! is too large for most floating-point variables. Your task is to find the rightmost non-zero digit of n!. For example, 5! = 1 * 2 * 3 * 4 * 5 = 120, so the rightmost non-zero digit of 5! is 2. Also, 7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040, so the rightmost non-zero digit of 7! is 4.

The first line of the input contains the number of test case (<15). The single line which represents each test case contains an integer n, between 1 and 1000 inclusive.

Print to the output the rightmost non-zero digit of n!

Input	Output
1 5	2

All-Ukrainian Collegiate Programming Contest
Semi-Final

April 8, 2010

Problem J

Zero

(1 sec)

Input: j.dat
Output: Standard Output

Let's consider the sequence of digits from 1 through N (where N=9) in increasing order: 1 2 3 ... N. Now insert either a '+' for addition or a '-' for subtraction or a ' ' [blank] to run the digits together, between each pair of digits (not in front of the first digit). Calculate the result that of the expression and see if you get zero.

Write a program that will find all sequences of length N that produce a zero sum.

The first line of the input contains the number of test cases (<10). Single test case is represented by single line that contains the integer N (3 <= N <= 9).

For each test case in ASCII order, print each sequence that can create 0 sum with a '+', '-', or ' ' (space) between each pair of numbers. Space shows that the digits run together. Different test cases should be separated by an empty line.

Input	Output
2 3 7	1+2-3 1+2-3+4-5-6+7 1+2-3-4+5+6-7 1-2 3+4+5+6+7 1-2 3-4 5+6 7 1-2+3+4-5+6-7 1-2-3-4-5+6+7