ACM-ICPC Asia Phuket Regional Programming Contest 2015

Hosted by
Department of Computer Engineering
Prince of Songkla University
Phuket Campus

20 November 2015

Contest Problem Set

- There are 12 problems (A-L) to solve within 5 hours (300 minutes).
- Solve as many problems as you can, in an order of your choice.
- Use C or C++ or Java to program at your convenience for any problems.
- Input and output of each program are **standard input** and **output**.

Problem A	Hex Code
Problem B	Contest Strategy
Problem C	Terrorists
Problem D	Aquarium
Problem E	Queue of Soldiers
Problem F	Jumping Joey
Problem G	Primorial vs LCM
Problem H	Automatic Scholarship Calculation
Problem I	Tom and Jerry
Problem J	Sunlight on a Tree
Problem K	Angry Birds
Problem L	Donation Packaging

Problem A. Hex Code

Time Limit: 1 sec

In the movie *The Martian* (2015), astronaut Mark Watney, one of the crew members of Mission Ares III, was left behind on Mars due to an unexpected incident during the surface exploration on the planet Mars. The communication with Earth was quasi-inexistent. Fortunately, Mark Watney managed to establish a very simple way to communicate with NASA at the mission control base on Earth through hexadecimal codes.

Mark could receive one simple code at a time which he could detect as a hexadecimal digit or hex code including `0-9` and `A-F`. Then he could transform *a pair of hex digits* into *a character* since he knows how to look up the ASCII table. Fortunately enough he has an ASCII table at hand (who would miss an ASCII table on voyage to Mars!! ^_^).

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

If Mark receives a code sequence "`4869204D61726B`", he can decode it into the message "`Hi Mark`". However, decoding the hex codes manually is very slow and inefficient. Therefore your task is to help Mark Watney write a program to decode these hex codes.

Input

The input contains several lines of hex codes only (`0-9`, `A-F`). Each line contains an *even number* of hex digits that you have to transform into a plain text message. One pair of hex digits corresponds to a single character. There are less than 250 hex digits in each line. The input contains several lines of strings. You do not know in advance how many lines there are but it is guaranteed that the input contains less than 1,000 lines.

Output
For each line of hex codes, print out the corresponding text message.

Sample Input	Sample Output
4869204D61726B 41726520796F7520616C6976653F	Hi Mark Are you alive?

Problem B. Programming Contest Strategy

Time Limit: 1 sec

As you are a contestant participating in an ACM-ICPC contest, your goal is to solve as many problems as you can within the competition duration with the least total time possible.

Let's suppose that you can, in advance, estimate the time (in minutes) that you will spend to solve each problem. Your task is to plan how to solve the contest problems that can maximize the number of solved problems while minimize the total time. The total time is the sum of the time (in minutes) for each problem solved at the submission time which is the elapsed time (in minutes) from the start of the contest. In your plan, at best you can assume that you can solve every problem at the first submission on the estimation time, so you do not have to worry about the penalty time. You can only work on 1 problem at any time (no parallel work).

Input

The first line contains integer T ($1 <= T <= 20$) which is the number of test cases. Each test case has 2 lines. The first line of each test case contains 2 integers: N ($1 <= N <= 20$) is the number of contest problems and L ($1 <= L <= 1{,}500$) is the duration of the contest (time in minutes). Then the second line contains N integers indicating the estimation time for each contest problem.

Output

For each test case, print out the case number followed by your best result possible containing 3 integers. The first value is the number of solved problems. The second value is the time for your last solved problem and the third value is the total time. See the samples for the exact format of output.

Sample Input	Sample Output
2 6 100 15 23 41 12 15 20 5 200 23 45 35 49 28	Case 1: 5 85 228 Case 2: 5 180 471

Explanation:

For the first test case in the sample input, the contest has 6 problems (A-F) and the competition lasts 100 minutes. The times that you will use for each problem are given in the second line, i.e. Problem A uses15 minutes, Problem B uses 23 minutes, Problem C uses 41 minutes and so on.

For the best score in this particular contest, you will solve at most 5 problems. The last problem that you can solve is submitted at 85 minutes and will have the total time of 228. You will not have time to solve problem C.

Problem C. Terrorists

Time Limit: 5 sec

Terrorists! There are terrorists everywhere!!! I was shouting out loud after I had watched a few news reports about terrorism acts that had happened around my neighborhoods. I started to think that hiding at home wasn't a good way to go.

I went to police stations and asked around if I could help them prevent these issues. The police gave me pieces of information about terrorists' plans. I ended up lying on my bed and figuring way to utilize this information. That is why I come to you.

Our neighborhoods could be illustrated with intersections and roads. Terrorists usually meet up at one intersection before moving to another intersection to perform an illegal act. The given information tells us where they will meet and where they will go. Unfortunately, the certain schedules of those plans are not available and too few policemen are available lately. So, the police will not be able to set up efficient defenses to all of those acts. What they could do is set up surveillance cameras to all meet up intersections. Once a meet up is detected, policemen will go to that meet up's destination to catch the terrorists. The policemen rarely accomplish it because they spend too long time travelling between places.

Because terrorists are smart, they always use shortest route to travel between intersections. Because the policemen aren't that smart, they need our help. For each terrorist plan, the police want us to compute the shortest distance between the meet up place and the destination. If the distance is too short, they will not spend their efforts for free.

Input

The first line of the input contains an integer T, the number of test sets ($1 <= T <= 5$).

Each test case consists of many lines. The first line contains 3 integers N, M, Q which are the number of intersections, the number of roads, and the number of terrorists' plans in respective order. $1 <= N <= 100\,000$, $N - 1 <= M <= N + 50$, $1 <= Q <= 50\,000$

Then the next M lines describe the roads in our neighborhoods. A road is described by 3 integers: U, V, D. U_i and V_i represent 2 ends of the road *i*. D_i represents distance of that road. $1 <= U_i, V_i <= N$, $1 <= D_i <= 10\,000$. It is possible that many roads might exist between a pair of intersection.

Finally the next Q lines are the terrorism plans. A plan *j* is described by 2 integers: S_j and E_j which are the meet-up intersection and the destination intersection respectively. $1 <= S_j, E_j <= N$.

Output

For each test set, print out "Case *x*:" where *x* is a case number beginning with 1, and then followed by Q lines. For each terrorism plan, output the shortest distance between the meet-up intersection and the destination of the plan.

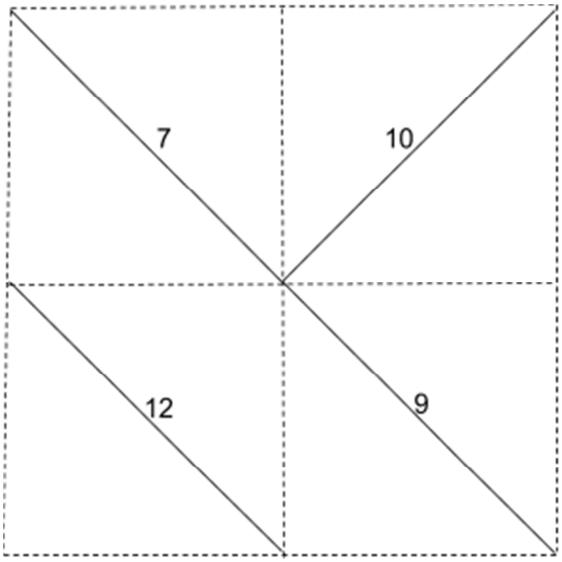
Sample Input	Sample Output
1 7 9 5 7 6 6114 5 3 5473 2 4 2601 5 4 8525 7 3 291 2 7 3363 1 6 399 6 4 4477 1 7 3075 6 3 4 4 3 1 2 3 7 3	Case 1: 3765 0 3366 3654 291

Problem D. Aquarium

Time Limit: 3 sec

There were different types of fish in your aquarium. But they did not go along well with each other. So there had been *Fish-War-1* among them. It was a complete mess. Lot of fishes died, many of them hid in some mountain, some were eaten by other fishes and so on. So you decided to compartmentalize your aquarium. You divided your aquarium into R x C grids, that is R rows and C columns. Then you inserted walls into each cell. The walls are slanted, that is it goes from north-east corner to south-west corner or north-west corner to south-east corner. They look like "/" or "\" respectively. Many years passed since the war. Now the fishes want to unite again. They want to bring down the walls. They measured the strength of each of the walls. What is the minimum amount of strength they need to spend to unite all the compartments?

For example, in the following 2 x 2 grid, they can spend 7 + 9 + 12 = 28 unit strength to unite the 4 compartments. And this is the minimum.



Input

First line of the input contains number of test case T (<= 20). Hence follows T test cases.

First line of the test case describes number of row R and number of columns C (1 <= R, C <= 100). Next R lines describe the walls. Each of these lines contains C characters and the characters are either "/" or "\". Next R lines contain C positive integers, each describes the strength of the wall at the corresponding cell. The strength of a wall would be at most 10,000.

Output

For each test case output the case number and the minimum amount of strength to unite all the compartments in the aquarium.

Sample Input	Sample Output
2 2 2 \/ \\ 7 10 12 9 1 3 /\\ 3 4 5	Case 1: 28 Case 2: 12

Problem E. Queue of Soldiers

Time Limit: 4 sec

In a planet far away from Earth, there is a deadly war going on between two countries: Bitland and Byteland. The queen of Bitland has an army of **N** soldiers. She has found a strategy to attack the weakest part of the border of Byteland, but to reach there, the soldiers of Bitland has to form a queue and pass through a very narrow valley. Moreover, there is some defense mechanism set in the valley by Byteland: If any soldier P with height **v** passes through the valley, then anybody following P with height (strictly) greater than **v** will be killed by automatic firing.

The queen of Bitland is also aware of this scenario, but it is not always possible to send her soldiers according to the non-increasing order of heights (so that no soldier dies).She understands that some soldiers have to sacrifice their lives. To find the best possible permutation, she wants your help to calculate the number of different permutations of the heights of her soldiers so that exactly **K** soldiers die in that valley.

A permutation is X different from another permutation Y if there exists some **i (1 ≤ i ≤ N)** for which the height of i-th soldier is different in these two permutations.

Input

The first line of input file contains the number of test cases, **T (1 ≤ T ≤ 20)**. Then **T** cases follow:

Each case consists of two lines. The first line contains two integers: **N (1 ≤ N ≤ 50000)** and **K (0 ≤ K ≤ 1000)**. Then the second line contains **N** integers denoting the heights of the soldiers. There will be at most **100** different values of their heights. There will be at most **1000** soldiers with same height.

For **80%** of the cases, **N** will be less than **10000**.

Output

For each case, print "**Case <x>: <y>**" in a separate line, where **x** is the case number and **y** is the number of permutations such that exactly **K** soldiers die. As the number can be very large, print **y modulo 1000000007**.

Sample Input	Sample Output
3 3 1 1 2 3 3 1 1 2 2 3 2 1 2 3	Case 1: 3 Case 2: 1 Case 3: 2

Problem F. Jumping Joey

Time Limit: 3 sec

Please read the problem statement carefully. Mathematical notations and bunch of examples are provided to make the statement friendly.

Once upon a time, there lived a frog named Joey. He has a long pond beside his house. There are lots of lily pads there, and he likes to jump from one pad to another. He hates to wet himself. Let's help Joey to cross the pond.

For the sake of simplicity, let's assume the pond to be a line segment of length **L**. On this line segment there are **n** lily pads. Let's enumerate the lily pads from left to right, that is the leftmost pad is number **1**, second one from the left is number **2** and so on. At the beginning Joey is at the left end of the pond. Then he moves to the first pad, then to the second pad and so on. At the end he moves from the nth pad to the right end of the pond. There are two ways he can move, jump and swim. He can jump from one place to another if the distance between these two places is no more than **D** unit. He can swim any times he wants. But as we already said he does not like to wet himself, so we need to minimize the number of times he swims for going from the left end to the right end.

However, the situation is not that simple. There are ropes between every two adjacent places. That is, there is rope between:

a. Left end and first pad
b. Last pad and right end
c. Between every two adjacent pads

Let,

a. $\mathbf{P_i}$ be the **i**'th pad (**1 <= i <= n**), $\mathbf{P_0}$ be the left end and $\mathbf{P_{n+1}}$ be the right end.
b. $\mathbf{r_i}$ be the length of the rope $\mathbf{R_i}$ between $\mathbf{P_i}$ and $\mathbf{P_{i+1}}$ (for **0 <= i <= n**).
c. $\mathbf{p_i}$ be the position of $\mathbf{P_i}$ (**0 <= i <= n + 1**). Obviously $\mathbf{p_0 = 0}$, $\mathbf{p_{n+1} = L}$ and $\mathbf{p_i < p_{i+1}}$, $\mathbf{r_i >= p_{i+1} - p_i}$ (for **0 <= i <= n**).

When Joey is on $\mathbf{P_i}$ he can pull $\mathbf{R_i}$ and then $\mathbf{P_{i+1}}$ moves towards Joey. If the rope $\mathbf{R_{i+1}}$ is taut (the length of the rope is equal to the distance between the two pads that the rope is tied with) then this also affects $\mathbf{P_{i+2}}$ and $\mathbf{P_{i+2}}$ moves toward him as well (and so on). However, if a rope is not taut then it does not affect later pads. Also if the ropes are taut till $\mathbf{P_{n+1}}$ then there is no movements of the pads at all (since the right end is fixed, that is $\mathbf{P_{n+1}}$ is not movable). Let's try to clarify these statements by some examples.

Example 1: Let Joey is on $\mathbf{P_2}$, $\mathbf{p_2 = 10}$, $\mathbf{p_3 = 20}$, $\mathbf{p_4 = 30}$, $\mathbf{p_5 = 40}$. Also $\mathbf{r_2 = r_3 = r_4 = 15}$. Distance between $\mathbf{P_2}$-$\mathbf{P_3}$, $\mathbf{P_3}$-$\mathbf{P_4}$ and $\mathbf{P_4}$-$\mathbf{P_5}$ are **10**, but the rope lengths are **15**. So none of the ropes are taut. Now if Joey pulls $\mathbf{R_2}$ say by **1** unit, $\mathbf{P_3}$ will shift one unit towards $\mathbf{P_2}$ (new $\mathbf{p_3 = 19}$). But this does not affect $\mathbf{p_4}$, because $\mathbf{R_3}$ was not taut. Let Joey pulls rope $\mathbf{R_2}$ **4** more units (**5** units in total). Then $\mathbf{p_3 = 15}$, $\mathbf{p_4 = 30}$, $\mathbf{p_5 = 40}$. Now $\mathbf{R_3}$ becomes taut since $\mathbf{P_3}$-$\mathbf{P_4}$ distance is now: $\mathbf{p_4 - p_3 = 30 - 15 = 15}$ which is same as $\mathbf{r_3}$. So now, if Joey pulls one more unit, $\mathbf{P_3}$ and $\mathbf{P_4}$ moves together ($\mathbf{P_5}$ does not, since $\mathbf{R_4}$ is not taut). So the new positions would be: $\mathbf{p_3 = 14}$, $\mathbf{p_4 = 29}$, $\mathbf{p_5 = 40}$.

Example 2: Let Joey is on P_0 and **n = 1**. Position of the only lily pad P_1 is at p_1 **= 10**. Let pond length **L = 20**. By definition p_0 **= 0** and p_2 **= L = 20**. Also let's assume r_0 **= 10,** r_1 **= 11**. If Joey pulls R_0 by one unit then: p_1 **= 9**. But now R_1 is taut. If he pulls R_0 more P_1 will not move, because the rope between P_1 and P_2 is taut and P_2 is the right end.

Given all the information, you have to figure out minimum number of times Joey has to wet himself in order to go from the left end to the right end. Please note Joey has to move from one place to the next place (he can't move backward), so he can't just wet himself once and swim from the left end to the right end.

Input

First line of the input will be a positive integer **T** (**<= 50**), number of test cases. Test cases are separated by blank lines. Each case consists of 3 lines. First line contains two positive integers **n** (**<= 1,000**) and **D** (**<= 10^9**). Second line contains **n + 2** integers: p_0, p_1, … p_{n+1}. Third line contains **n + 1** positive integers r_0, r_1 … r_n. None of the p_i and r_i will be more than **10^9**. You may assume that all of these given values will satisfy all the constraints in the statement. Please note there may be one or more spaces/new lines separating adjacent lines/numbers.

Output

For each of the tests print the case number and the answer.

Sample Input	Sample Output
5 1 10 0 10 20 10 10 1 10 0 11 20 11 10 1 10 0 11 20 11 9 1 5 0 10 20 20 20 1 5 0 10 20 20 12	Case 1: 0 Case 2: 0 Case 3: 1 Case 4: 1 Case 5: 2

Problem G. Primorial vs LCM

Time Limit: 4 sec

Given N ($2<=N<=10^{14}$), what is the quotient of LCM(1,2,3,....,N) divided by multiple of all primes up to N. As the result might be too big, output it's modulo by 1000000007.

For example, when N=5, the result is LCM(1,2,3,4,5)/(2*3*5)=60/30=2.

Note that LCM stands for Lowest or Least Common Multiple.

Input

The first line of the input is T(T ≤ 50000), then T test cases follows in next T lines. Each line contains an integer N (2 ≤ N ≤ 100000000000000 or 10^{14}). The meaning of N is given in the problem statement.

Output

For each test case print a line in "Case *x*: *S*" format where *x* is case number and *S* is the quotient modulo by 1000000007.

Sample Input	Sample Output
10 2 3 4 5 6 7 8 9 10 1000	Case 1: 1 Case 2: 1 Case 3: 2 Case 4: 2 Case 5: 2 Case 6: 2 Case 7: 4 Case 8: 12 Case 9: 12 Case 10: 744593350

Problem H. Automatic Scholarship Calculation

Time Limit: 4 sec

A university gives scholarship to students based on their current CGPA. This waiver allocation follows some rules. At the beginning of a semester allotted scholarships for certain CGPA range is put on the notice board in tabular form. Two such tables are shown below:

CGPA Range	Scholarship Amount
4.00-4.00	50%
3.90-3.99	40%
3.80-3.89	30%
3.70-3.79	20%
3.60-3.69	10%

CGPA Range	Scholarship Amount
3.91-4.00	80%
3.71-3.90	50%
3.51-3.70	20%

In the table on the left it is said that students who have CGPA within 3.60 and 3.69 (inclusive) will get 10% of their tuition fee as scholarship. Similarly, those with CGPA range is within 3.90-3.99 will get 40% of their tuition fee as scholarship. Using the two tables above we will try to explain to you the rules and restrictions of allocating waiver:

1. A fixed Scholarship is given to all the students within a CGPA Range.
2. The range of each CGPA slab must be equal (Except the topmost one, which can be smaller). The slabs must go all the way up to 4.00.
3. The scholarship percentage for the lowest slab is a positive integer. With the increase of CGPA range the amount of scholarship percentage must also increase by a fixed positive integer value. In the table on the left this fixed value is 10% and in the table on the right this fixed value is 30%.
4. Scholarship percentage for each slab is always a positive integer with a maximum value of 100%.
5. The scholarship amount that can give 1 student 1 % scholarship is called a unit. So to give 2 students 50% scholarship, 50*2 = 100 units is needed.
6. All the scholarships given should use up a given amount, P units.
7. A CGPA range for scholarship cannot start below 2.50. So 2.50-2.55 is a valid CGPA range but 2.45-2.55 is not valid.
8. There must be at least 2 slabs for scholarship. But of course a slab can contain zero students.

Given the number of students, total available scholarship units and their current CGPA, your job is to find out the number of different possible scholarship allocations that uses up all the scholarship units. Two scholarship allocations are different if their CGPA range is different or scholarship allocation in any slab is different.

Input

Input file contains at most 100 sets of inputs. The description of each set is given below:

Each set starts with two integers *N* (N<=10000) which indicates the total number of students and *P* which indicates the scholarship units that needs to be used up. Each of the next *N* lines contains the CGPA of one student.

Input is terminated by a line containing two zeroes.

Output
For each test case produce one line of output which contains an integer D. This D denotes the total number of different scholarship allocations that uses up all P units. Input will be such that there will always be at least one possible allocation.

Sample Input	Sample Output
10 410 3.29 3.96 2.61 2.82 3.93 3.68 3.70 2.91 3.28 3.68 0 0	79554

Illustration:
//a = 10 d = 10 lowest = 2.88 nslab = 8 wslab = 14
One possible solution to the sample input above is

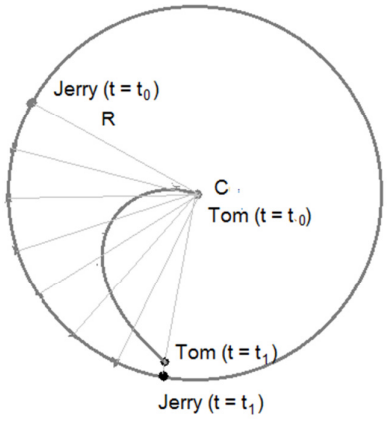
CGPA Range	# of students	Waiver %	Total Units needed
3.93-4.00	2	80	160
3.78-3.92	0	70	
3.63-3.77	3	60	180
3.48-3.62	0	50	
3.33-3.47	0	40	
3.18-3.32	2	30	60
3.03-3.17	0	20	
2.88-3.02	1	10	10
		Total Units	410

There is another 79553 different allocation which uses up all 410 units exactly.

Problem I. Tom and Jerry

Time Limit: 2 sec

Tom and Jerry are very fond of cat and mice games, which might be rather obvious to you. Today they are playing a very complicated game. The goals are simple as usual though, Jerry would be running and Tom would have to catch Jerry.



However, today Jerry is running on a perfect circular path with radius R meters, at a constant speed of V m/s. Initially Tom is sitting at the very center of the circle. He wants to catch Jerry as soon as possible, but we all know, Tom is not very intelligent. Instead of calculating an optimal direction to catch Jerry, he is just running towards Jerry.

As Jerry is also moving, the path Tom has taken start to look like a curve (see picture above). At any given moment, Tom's position is between Jerry's current position and the center of the circle. Tom is also moving at a constant speed of V m/s, same speed as Jerry. Find the time (in seconds) Tom would need to catch Jerry.

Input

Input file has T (T <= 10000) test cases, each case consists of two integer R and V. Here, 0 < R, V <= 10000.

Output

For each test case, print the case number and the time Tom will need to catch Jerry. Floating point rounding error lower than 1e-5 will be ignored by the judge.

Example

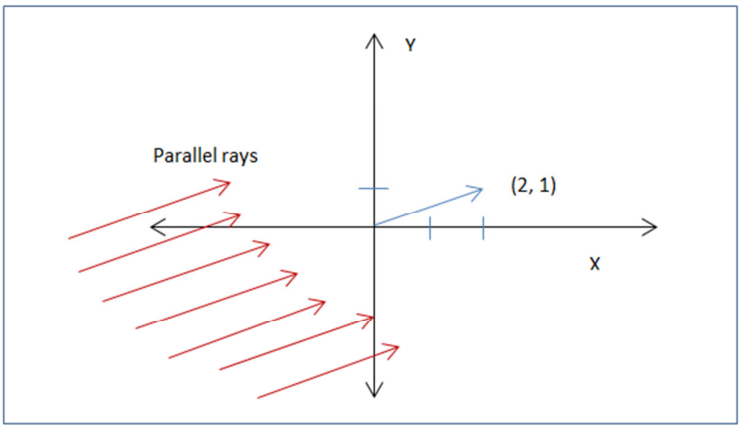
Sample Input	Sample Output
4 45 100 5 1547 1000 10000 5668 5467	Case 1: 0.70685835 Case 2: 0.00507691 Case 3: 0.15707963 Case 4: 1.62854830

Problem J. Sunlight on a Tree

Time Limit: 5 sec

Let's take the top view of a tree. If you consider tree branches to be edges and the places where branches meet or end to be nodes, then it will be an embedding of a graph in 2D plane (not necessarily planar). Actually this graph is a tree. That is, the graph is connected but contains no cycle. In other words, this graph is connected and has *n* nodes (2D coordinates) with *n - 1* edges.

Now you will be given a bunch of queries. Each query consists of 4 numbers "u v x y". u and v refers to two nodes. (x, y) is the vector that indicates the direction of sunlight. The figure below shows an example of (x, y) = (2, 1). There is the sunlight coming from the lower left and the rays are parallel with the vector (2, 1). Please note that the sunlight is not a single ray. Rather it is a bunch of parallel rays moving from infinite towards a certain direction.



We think that a node is warmer if it is closer to sun. If an ant moves from node u to node v in the unique shortest path, which node(s) will be the warmest location for the ant?

Let's give an example. Consider the following tree (see the figure next page) and a few queries:

a. u = 14, v = 4, x = 1, y = 0: The unique shortest path from u = 14 to v = 4 is: [14, 11, 1, 3, 4]. (x, y) = (1, 0) means sunlight is coming from left and going towards right. In this case, node 3 and 11 are closest to sun. So the output will be 3 and 11.
b. u = 13, v = 9, x = 1, y = -1: The unique shortest path from u = 13 to v = 9 is: [13, 11, 1, 6, 9]. (x, y) = (1, -1) means sunlight is coming from upper left corner and going towards lower right corner. In this case node 11, 1 and 6 are closest to sun. So the output will be: 1, 6 and 11.

Input

First line of the input will contain number of test cases T (T ≤ 10). Hence follows T test cases. Each test case starts with n, number of nodes in the tree (n<= 10^5). In the following n lines, there will be n 2d coordinate points (|x|, |y| <= 10^5). Next n - 1 lines will describe the edges of the tree by giving id (1 <= id <= n) of the two nodes that an edge connects. You may assume that the input will be a valid tree. In the next line a positive integer Q will be given (<= 1.6 * 10^5) denoting number of queries. In the next Q lines, each query will be of the form "u v x y" (1 <= u, v <= n and |x|, |y| <= 10^5). Please note all the cases are not worst case but there is at least one case with the highest possible number of nodes and queries.

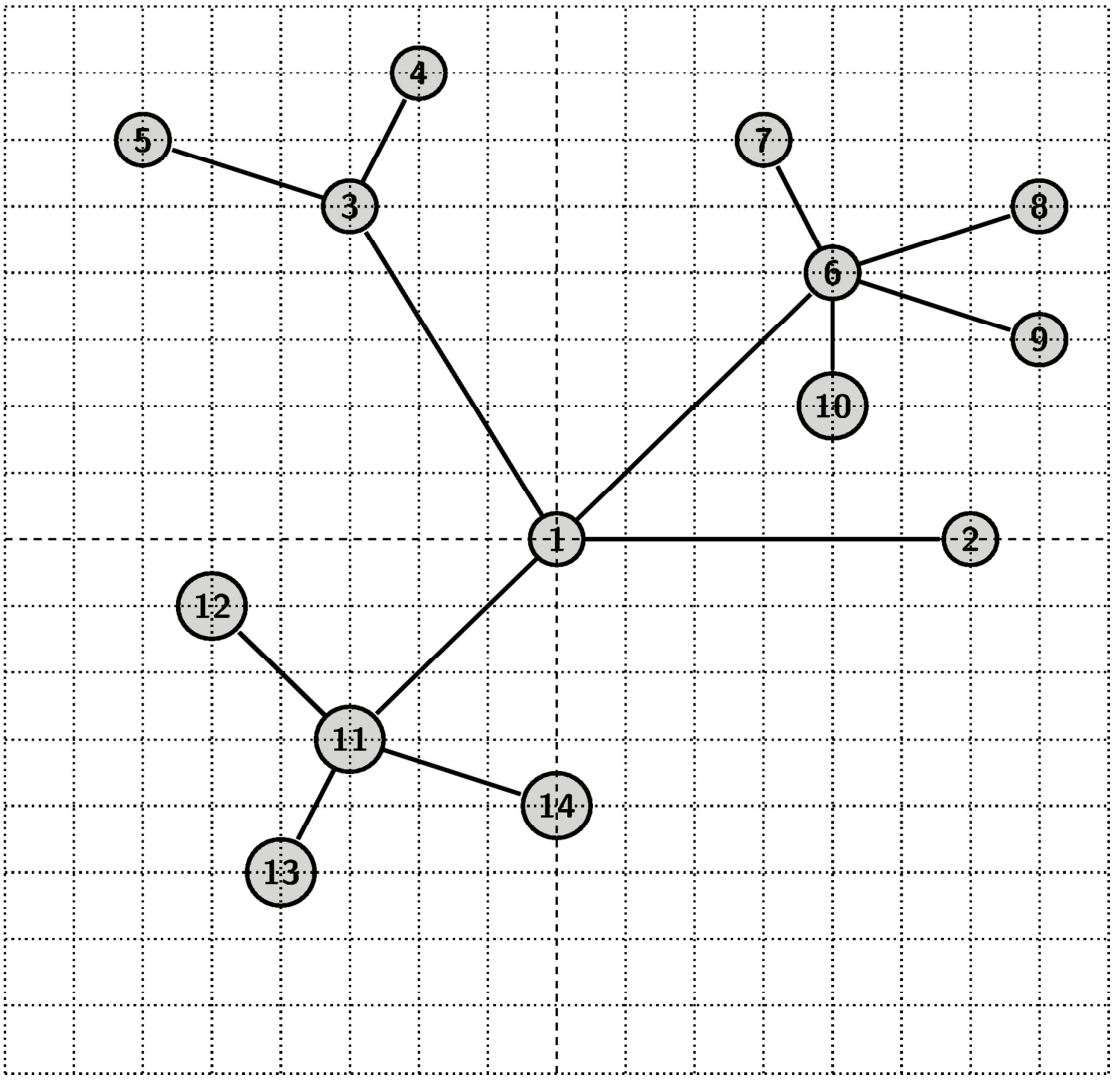


Figure J. Example Tree

Output

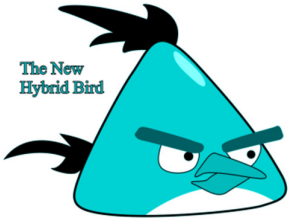
For each test case print the case number, followed by the ids of the nodes that are hit by the sunlight first. If there are multiple points hit by the sunlight first, sort them by their id and print them in space separated way. Please do not put any space at the end or beginning of the line. You may consider that in total your code will print at most 3*10^5 ids.

(Please see the sample input/output next page)

Sample Input	Sample Output
2 4 0 0 1 1 0 1 1 0 1 2 1 3 1 4 3 2 3 0 -1 2 3 0 1 2 3 1 -1 14 0 0 6 0 -3 5 -2 7 -6 6 4 4 3 6 7 5 7 3 4 2 -3 -3 -5 -1 -4 -5 0 -4 1 2 1 3 3 4 3 5 1 6 6 7 6 8 6 9 6 10 1 11 11 12 11 13 11 14 2 14 4 1 0 13 9 1 -1	Case 1: 2 3 1 3 Case 2: 3 11 1 6 11

Problem K. New Angry Bird Weapon

Time Limit: 3 sec



One of the most popular games for the Android, IOS world is Angry Birds which is developed by Rovio Entertainment Limited. Many different sorts of weapons are used in this game. Three of them have color red, yellow and blue. The red one has a conventional projectile path. The yellow one also has conventional projectile path but after throwing the weapon when the screen is touched again the yellow one continues to hold its line with a slight change in direction (The effect of g is almost nullified and shown in figure 1). On the other hand the blue one also has a projectile path when thrown but splits itself into three parts when the screen is touched again. The middle part continues in the original path of projectile while the other two goes slightly left or right (Shown in figure 2). In this problem we will introduce a new angry bird weapon which is a hybrid of these two (Yellow and Blue Weapon).

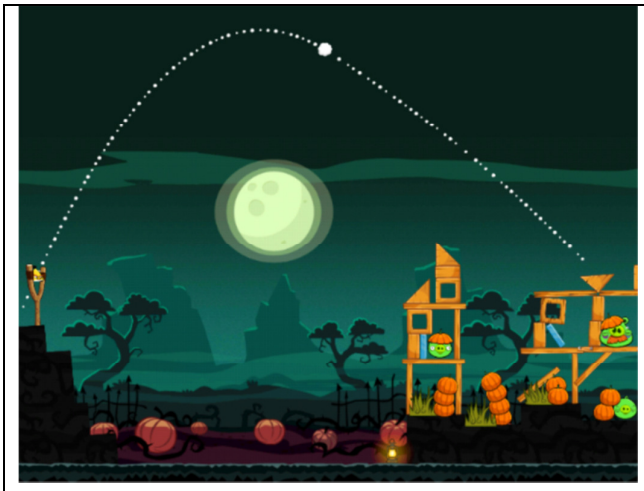


Figure 1: Screen is touched when the yellow weapon reaches the location marked with a white circle.

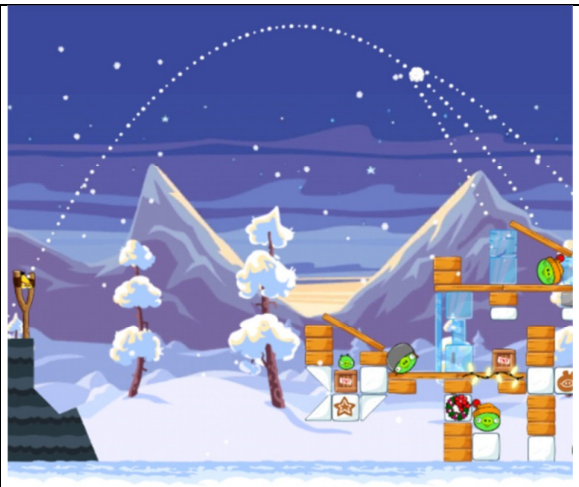


Figure 2: Screen is touched when the blue weapon reaches the location marked with a white circle.

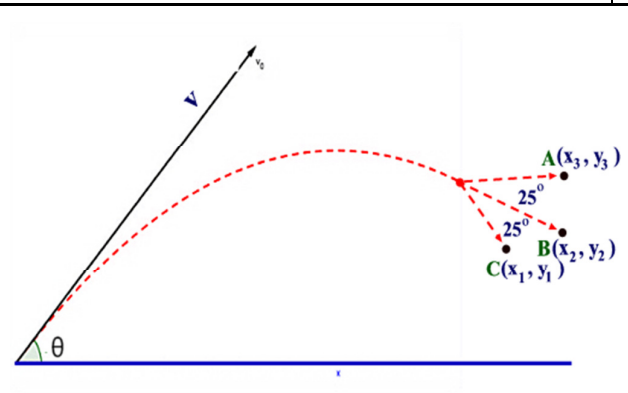


Figure 3: Paths of the new weapon

Our new weapon initially has the path of a projectile. But when screen is touched it splits itself into three parts and each of them continues in a straight line path. The middle part holds the direction of the projectile and continues in a straight-line. Other two parts make 25 degree angle with this direction on left and right side and continue in a straight line (As shown in figure 3). Finally those three parts hit three targets A, B and C at coordinates (x_1,y_1), (x_2, y_2) and (x_3,y_3).

Now in this problem you will be given the location of three targets and the weapon is always thrown from the origin (0,0). You will have to find the angle θ (expressed in degree) in which the weapon has to be thrown to hit the targets.

Input

First line of the input contains an integer T (T<=10000) which denotes how many test cases there are. Each of the next T lines contains a single test case. The description of each line is given below.

Each line contains six floating-point numbers x_1, y_1, x_2, y_2, x_3 and y_3 (500.0 <= x_1, x_2, x_3 <= 10000.0) which denotes that the coordinate of three targets are (x_1, y_1), (x_2, y_2) and (x_3, y_3) respectively. The minimum distance between any two targets is 5 units.

Output

For each query produce one line of output. This line contains the serial of output followed by a floating-point number θ which denotes the angle (expressed in degree) in which the weapon needs to be thrown to make it able to hit all targets after exploding. This floating-point number should have six digits after the decimal point. If there is more than one solution, output the one that needs lesser initial speed to go along with it to hit the targets. Look at the output for sample input for details. You can assume that the weapons does not hit the target before splitting (even if it goes through the target) and there is no effect of wind or anything else on the weapon except the effect of g. But after splitting there is no effect of g on it as well. You can also assume that there will always be a solution (A valid value for θ). You must also assume that for a valid solution the throwing speed should be minimum 50 and maximum 5000 units and the throwing angle is in between 10 and 80 degree (Any solution that is not within this limit is also considered invalid). You should assume that g = 9.8 units/s^2.

Sample Input

```
2
6509.57475927 1176.42632622 6851.46336994 1279.22002747 7151.04630423 1589.26357354
4904.79101589 727.73576687 4782.48141525 1064.34391907 4699.50241228 426.80332837
```

Sample Output

```
Case 1: 49.329940
Case 2: 12.243520
```

Problem L. Donation Packaging

Time Limit: 1 sec

ICPC donation center is preparing donation packages for people who faced disaster in every area in Thailand. The center gets several kinds of donation. In this year, the center gets a request to pack the basic items in one package which consists of three sets. The first set, called *SetA*, is water and beverage. The second one, *SetB*, consists of instant noodle, canned food and rice. And the last set, *SetC*, is first aid kits and tissue paper. In every day, we try to pack and distribute dotation packages as many as possible. We have constrains for packaging which are: one package has to comprise of SetA , SetB and SetC equally and one distribution contains the minimum of 30 packages. However, the center receives these 3 sets in different amount each day. Therefore, the center must wait until the accumulated amount of each and every set over 30 in order to pack a distribution.

Input

The input will start with an integer ***T*** (1<= ***T*** <= 365), the number of consecutive donation days. The next ***T*** lines show donation in each day, consisting of three integers: the number of *setA*, *SetB* and *setC*, respectively.

Output

For each day, print out the maximum number of packages to be distributed on that day. If the packages cannot be distributed on that day (number of packages is less than 30), print NO.

Sample Input	Sample Output
5 20 30 60 20 30 60 50 40 60 50 10 10 0 50 50	NO 40 50 NO 50