The 2009 ACM Asia Programming Contest
Wuhan Site

sponsored by IBM
hosted by Wuhan University
Wuhan, China

November 1, 2009

This problem set should contain ten (10) problems on fifteen (15) pages.
Please inform a runner immediately if something is missing from your problem set.

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem A
Assembling Services
Input: assembling.in

In this problem, you need to simulate the execution of n service programs P_1, P_2, ..., P_n. Each program is described with a sequence of integers: T I in_1 in_2 ... in_I O out_1 out_2 ... out_O, that means it takes T unit time to execute, needs I input variables (i.e. in_1 in_2 ... in_I), and sets O output variables (i.e. out_1 out_2 ... out_O) when it finishes running. A program can be started if and only if all these T input variables are ready (initially available, or set by some other programs).

Imagine you have a super-computer which can execute as many programs in parallel as you like, and every variable can be read and written simultaneously by multiple programs. Your task is to calculate a particular "target" variable, as soon as possible.

Assume there are 4 programs, shown in the table below:

Program No.	Time	Requires	Produces
1	2	X_1	X_2
2	3	X_1	X_3
3	4	X_2	X_4
4	1	X_3, X_4	X_5

The quickest time to get X_5 is 7, if only X_1 is available at startup.

You also need to construct an expression that shows **how to execute the programs to achieve the minimal time**. The grammar of the expression is recursive:

- **Single Program**: `Px`, where $1 \le x \le n$. (i.e. P2, P499, etc). Meaning: execute the program immediately. Then end of this program marks the end of this expression.
- **Execute in serial**: `(S1S2...Sk)`, where every `Si` is an expression. Note that the outermost pair of parentheses is mandatory. Meaning: execute expression `S1`, then `S2` immediately after `S1` ends, then `S3` immediately after `S2` ends, ..., and finally `Sk` immediately after `Sk-1` ends. Then end of expression `Sk` marks the end of the whole expression.
- **Execute in parallel**: `(S1|S2|...|Sk)`, where every `Si` is an expression. Note that the outermost pair of parentheses is mandatory. Meaning: execute expressions `S1`, `S2`, ..., and `Sk` simultaneously. The end of last finished expression marks the end of the whole expression.

One of the possible expressions for the example above is `(((P1P3)|P2)P4)`. `(P1P2P3P4)`is not acceptable, since X_5 is available at time 10 in that expression, later than the optimal time 7.

Input

There will be at most 100 test cases. Each case begins with three integers n, m, o($1 \le n,m \le 500$, $1 \le o \le m$). The number of programs is n, the number of variables is m, and the target variable is X_o. Variables are numbered 1 to m, programs are numbered 1 to n. The next line contains a 01 string of m characters. The i-th character is 1 if and only if the i-th variable is initially available. The target variable is guaranteed to be unavailable at startup. The following n lines describe the programs. Each line begins with an integer T($1 \le T \le 100$), the execution time, and an integer I followed by I integers in_1, in_2, ..., in_I, as stated above, then an integer O followed by O integers out_1, out_2, ..., out_O. $1 \le in_i, out_i <= m$, $1 \le I, O \le 10$. The last test case is followed by $n=m=o=0$, which should not be processed.

Output

For each test case, print the case number and the total time needed to get the target variable. If it's not possible to get the target variable, print `-1` in stead.

If it's possible to get the target variable, print the expression after that, in the same line. Be sure to print a valid expression having at most 10,000 characters, with each program printed at most once. There should be no whitespace characters within the expression.

To make this problem a little bit easier, it's allowed that some programs finish **after** the optimal time, as long as the target variable is available at the optimal time. You're also allowed to print redundant parentheses (pay attention to the expression length, though). If such an expression does not exist, print "`Can't do in serial-parallel.`", without quotes.

Print a blank line after the output of each test case.

Sample Input	Output for the Sample Input
4 5 5 10000 2 1 1 1 2 3 1 1 1 3 4 1 2 1 4 1 2 3 4 1 5 1 2 1 01 31 1 2 1 1 3 5 5 10100 3 1 1 1 2 1 1 3 1 4 3 2 4 2 1 5 1 3 3 100 1 1 1 1 2 0 0 0	Case 1: 7 (((P1P3)|P2)P4) Case 2: 31 P1 Case 3: 6 ((P1P3)|P2) Case 4: -1

Explanation

After a variable is set, it'll keep available forever. That's why `P3` can be executed, in the third example.

Also note that there are some other correct expressions for the first sample, e.g.`((P1P3P4)|P2)`. You can even print `(((P1P3)P4)|P2)` or `((P1(P3P4))|P2)`. Any one of them is acceptable in this problem.

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem B
Box Relations
Input: box.in

There are n boxes C_1, C_2, ..., C_n in 3D space. The edges of the boxes are parallel to the x, y or z-axis. We provide some relations of the boxes, and your task is to construct a set of boxes satisfying all these relations.

There are four kinds of relations ($1 \le i,j \le n$, i is different from j):

- `I i j`: The intersection volume of C_i and C_j is positive.
- `X i j`: The intersection volume is zero, and any point inside C_i has smaller x-coordinate than any point inside C_j.
- `Y i j`: The intersection volume is zero, and any point inside C_i has smaller y-coordinate than any point inside C_j.
- `Z i j`: The intersection volume is zero, and any point inside C_i has smaller z-coordinate than any point inside C_j.

Input

There will be at most 30 test cases. Each case begins with a line containing two integers n ($1 \le n \le 1{,}000$) and R ($0 \le R \le 100{,}000$), the number of boxes and the number of relations. Each of the following R lines describes a relation, written in the format above. The last test case is followed by $n=R=0$, which should not be processed.

Output

For each test case, print the case number and either the word `POSSIBLE` or `IMPOSSIBLE`. If it's possible to construct the set of boxes, the i-th line of the following n lines contains six integers x_1, y_1, z_1, x_2, y_2, z_2, that means the i-th box is the set of points (x,y,z) satisfying $x_1 \le x \le x_2$, $y_1 \le y \le y_2$, $z_1 \le z \le z_2$. The absolute values of x_1, y_1, z_1, x_2, y_2, z_2 should not exceed 1,000,000.

Print a blank line after the output of each test case.

Sample Input

```
3 2
I 1 2
X 2 3
3 3
Z 1 2
Z 2 3
Z 3 1
1 0
0 0
```

Output for the Sample Input

```
Case 1: POSSIBLE
0 0 0 2 2 2
1 1 1 3 3 3
8 8 8 9 9 9

Case 2: IMPOSSIBLE

Case 3: POSSIBLE
0 0 0 1 1 1
```

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem C
Crossing Rivers
Input: crossing.in

You live in a village but work in another village. You decided to follow the straight path between your house (A) and the working place (B), but there are several rivers you need to cross. Assume B is to the right of A, and all the rivers lie between them.

Fortunately, there is one "automatic" boat moving smoothly in each river. When you arrive the left bank of a river, just wait for the boat, then go with it. You're so slim that carrying you does not change the speed of any boat.

Days and days after, you came up with the following question: assume each boat is independently placed at random at time 0, what is the **expected time** to reach B from A? Your walking speed is always 1.

To be more precise, for a river of length L, the distance of the boat (which could be regarded as a mathematical point) to the left bank at time 0 is **uniformly chosen** from interval [0, L], and the boat is equally like to be moving left or right, if it's not precisely at the river bank.

Input
There will be at most 10 test cases. Each case begins with two integers n and D, where n ($0 \le n \le 10$) is the number of rivers between A and B, D ($1 \le D \le 1000$) is the distance from A to B. Each of the following n lines describes a river with 3 integers: p, L and v ($0 \le p < D$, $0 < L \le D$, $1 \le v \le 100$). p is the distance from A to the left bank of this river, L is the length of this river, v is the speed of the boat on this river. It is guaranteed that rivers lie between A and B, and they don't overlap. The last test case is followed by $n=D=0$, which should not be processed.

Output
For each test case, print the case number and the expected time, rounded to 3 digits after the decimal point.

Print a blank line after the output of each test case.

Sample Input	Output for the Sample Input
1 1 0 1 2 0 1 0 0	Case 1: 1.000 Case 2: 1.000

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem D
Download Manager
Input: download.in

Jiajia downloads a lot, a lot more than you can even imagine. Some say that he starts downloading up to 20,000 files together. If 20,000 files try to share a limited bandwidth then it will be a big hazard and no files will be downloaded properly. That is why, he uses a download manager.

If there are *T* files to download, the download manger uses the following policy while downloading files:

1. The download manager gives the smaller files higher priority, so it starts downloading the smallest *n* files at startup. If there is a tie, download manager chooses the one with less bytes remaining (for download). We assume that with at least 50 Mega Bytes/sec of bandwidth, *n* files can be downloaded simultaneously without any problem.

2. The available bandwidth is equally shared by the all the files that are being downloaded. When a file is completely downloaded its bandwidth is instantaneously given to the next file. If there are no more files left except the files that are being downloaded, this bandwidth is immediately shared equally by all remaining files that are being downloaded.

Given the size and completed percentage of each file, your task is to intelligently simulate the behavior of the download manager to find the total time required to download all the files.

Input
The will be at most 10 test cases. Each case begins with three integers *T* ($1 \le T \le 20000$), *n* ($1 \le n \le 2000$ and $1 \le n \le T$) and *B* ($50 \le B \le 1000$). Here *B* denotes the total bandwidth available to Jiajia (In Megabytes/sec). Please note that the download manager always downloads *n* files in parallel unless there are less than *n* files available for download. Each of next *T* lines contains one non-negative floating-point number *S* (less than 20,000, containing at most 2 digits after the decimal places) and one integer *P* ($0 \le P \le 100$). These two numbers denote a file whose size is *S* megabyte and which has been downloaded exactly *P*% already. Also note that although theoretically it is not possible that the size of a file or size of its remaining part is a fraction when expressed in bytes, for simplicity please assume that such thing is possible in this problem. The last test case is followed by $T=n=B=0$, which should not be processed.

Output
For each case, print the case number and the total time required to download all the files, expressed in hours and rounded to 2 digits after the decimal point. Print a blank line after the output of each test case.

Sample Input	Output for the Sample Input
6 3 90 100.00 90 40.40 70 60.30 70 40.40 80 40.40 85 40.40 88 1 1 56 12.34 100 0 0 0	Case 1: 0.66 Case 2: 0.00

Explanation
In the first sample, there are 6 files and the download manager can download 3 files simultaneously. The size of the smallest file is 40.40 Megabyte but there are four such files (2nd, 4th, 5th and 6th files). So the download manager chooses the 6th, 5th and 4th files for download as they have less bytes remaining. All these files get equal bandwidth (30.00 Megabyte/Sec). Of these three files the 8th file is finished first. So instantaneously the 2nd file starts downloading. Then, 5th file is finished. So the next larger file (3rd file) starts downloading. This process goes on until all files are downloaded.

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem E
Exclusive-OR
Input: exclusive.in

You are **not** given n non-negative integers $X_0, X_1, ..., X_{n-1}$ less than 2^{20} , but they do exist, and their values never change.

I'll gradually provide you some facts about them, and ask you some questions.

There are two kinds of facts, plus one kind of question:

Format	Meaning
`I p v`	I tell you $X_p = v$
`I p q v`	I tell you X_p XOR $X_q = v$
`Q k p1 p2 … pk`	Please tell me the value of X_{p1} XOR X_{p2} XOR ... XOR X_{pk}

Input
There will be at most 10 test cases. Each case begins with two integers n and Q ($1 \le n \le 20,000$, $2 \le Q \le 40,000$). Each of the following lines contains either a fact or a question, formatted as stated above. The k parameter in the questions will be a positive integer not greater than 15, and the v parameter in the facts will be a non-negative integer less than 2^{20}. The last case is followed by $n=Q=0$, which should not be processed.

Output
For each test case, print the case number on its own line, then the answers, one on each one. If you can't deduce the answer for a particular question, from the facts I provide you **before** that question, print "`I don't know.`", without quotes. If the i-th fact (don't count questions) **cannot** be consistent with **all** the facts before that, print "`The first i facts are conflicting.`", then keep silence for everything after that (including facts and questions). Print a blank line after the output of each test case.

Sample Input

```
2 6
I 0 1 3
Q 1 0
Q 2 1 0
I 0 2
Q 1 1
Q 1 0
3 3
I 0 1 6
I 0 2 2
Q 2 1 2
2 4
I 0 1 7
Q 2 0 1
I 0 1 8
Q 2 0 1
0 0
```

Output for the Sample Input

```
Case 1:
I don't know.
3
1
2

Case 2:
4

Case 3:
7
The first 2 facts are conflicting.
```

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem F
Final Combat
Input: final.in

You're playing an RPG, which is arguably the best Chinese RPG in 2007. You got to the final combat. You want to win as soon as possible. Do you know what I am talking about?

For competitors who're not familiar with the game: we're giving a brief introduction to the combat system and the characters involved in that last combat very soon.

For competitors who're too familiar with the game: we're simplifying and/or changing the rules and facts for this problem. We apologize if you feel unhappy about that.

Fig. The final combat

The screenshot above gives you some idea about the combat system and the characters, which can be summarized as follows:

- You have 4 heroes: Yun Tianhe (YTH), Han Lingsha (HLS), Liu Mengli (LML) and Murong Ziying (MRZY). Before the combat begins, you need to choose **exactly** three heroes and rearrange them in some order. Let's call the 1st, 2nd and 3rd hero H1, H2 and H3, respectively. In this screenshot above, H1 is YTH, H2 is HLS and H3 is LML. (Sorry for the Ziying fans!)
- There are two bosses: Xuan Xiao (XX) and Su Yao (SY), both in the back of the screenshot.
- The goal of this combat is to defeat SY. In this problem, you can assume that XX is immortal, although it's possible to defeat him in the real game.

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

The key to this problem is to understand the "semi" round-based combat system. See the "**progress bar**" on the top of the screen? Each hero has a very small **progress ball** (in light-blue) on the bar, with an icon above the ball. Similarly, each boss also has a ball (in red), but the icon is below the ball. When the combat begins, all the progress balls start moving from the left corner of the bar. When a ball reaches the right corner, the corresponding character is able to act. If more than one character can act, they make actions one by one, according to this precedence order: YTH, HLS, LML, MRZY, XX, SY. When a character is acting, all progress balls stopped moving. After a character finished an action, his/her progress ball is reset to the left corner of the progress bar. When nobody is acting, all progress balls go right simultaneously (but possibly with different speeds, see below).

In this problem, each character has 4 main properties: Jing, Qi, Shen, Su.

- **Jing** means "health". When a character's Jing reaches zero or negative, he/she is defeated. A character's maximum Jing is denoted by parameter *maxjing*.
- **Qi** means "vitality". It is used by special-skill attacks. When Qi is not enough, certain special-skill attacks cannot be performed. A character's maximum Qi is always 100.
- **Shen** means "spirit". It is used by Xian Shu (something similar to, but more amazing than "magic"). When Shen is not enough, certain Xian Shu cannot be performed. A character's maximum Shen is denoted by parameter *maxshen*.
- **Su** means "speed". It affects how quick a character's progress ball moves. Su is always a positive integer **less** than 5. If a character's Su is *x*, it takes 5-*x* units time for his/her progress ball to move from left corner to the right corner of the progress bar. A character's Su (which is never changed during the combat) is denoted by parameter *su*.

For simplicity, you may assume that both XX and SY use the same strategy:

- In his/her (4*n*+1)-th action (*n*=0,1,2,...), make a weapon attack upon H1.
- In his/her (4*n*+2)-th action (*n*=0,1,2,...), make a weapon attack upon H2.
- In his/her (4*n*+3)-th action (*n*=0,1,2,...), make a weapon attack upon H3.
- In his/her (4*n*+4)-th action (*n*=0,1,2,...), make a special-skill attack upon all the heroes.

Now we introduce four more parameters for each hero. The first two, *d1x* and *d2x*, are the damages that the hero takes on XX's weapon attack and special-skill attack, respectively. For SY's attacks, we define *d1s* and *d2s* similarly.

You're very tired, so you don't want to waste your time designing complex tactics. When a hero is about to act, you only consider the following actions:

- **Make a physical weapon attack** upon either XX or SY (cannot attack both). Be warned though, SY is surrounded by swords (see the screenshot), so making weapon attacks upon her would hurt the attacker himself/herself. The amount of "reflection damage" that the attacker takes is the same as the "physical damage" that SY takes. Attacking XX (yes, you **can** do that, if you like) does **not** suffer from physical reflection.
- **Use Xian Shu to recover Jing.** To make your brain easier, you decided **not** to recover other heroes' Jing. In this problem, the only Xian Shu that you can use is called "Yu Run", which recovers *yurun_jing* points of Jing (if the resulting Jing exceeds his/her maximum Jing, it is reduced to the maximum) and uses *yurun_shen* points of Shen.
- **Use items to recover Shen**. To make your brain easier, you decided **not** to recover other heroes' Shen. In this problem, the only item that you can use is called "Shu Er Guo", which recovers *shuerguo_shen* points of Shen (if the resulting Shen exceeds his/her maximum Shen, it is reduced to the maximum). You have an infinite number of Shu Er Guo.
- **Make a special-skill attack** if his/her Qi is enough. Each hero has exactly one special-skill attack, attacking both XX and SY. Note that some special-skill attacks are physical. Using a physical special-skill attack makes you take the same damage as SY, just like weapon attacks.

Now it's time to introduce four more parameters for each hero: *wad*, *ssd*, *ssq* and *ssp*. The number of Jing points that SY takes by the hero's weapon attacks and special-skill attacks are denoted by *wad* and *ssd*, respectively. The special-skill attack is physical if and only if *ssp*=1 (otherwise, *ssp*=0). It needs *ssq* points of Qi.

As you may have noticed, Qi cannot be recovered by Xian Shu or items. There are only two ways to increase Qi: make a weapon attack, or get hit by a weapon attack. Being hurt by physical reflection does **not** earn you extra Qi.

So here come the last two parameters for each hero: *q*1 and *q*2, that means making a weapon attack increases *q*1 points of Qi (whether or not you're reflected), while getting hit by a weapon attack increases *q*2 points of Qi. If the resulting Qi exceeds his/her maximum Qi (which is always 100), it is reduced to the maximum. Again, performing a special-skill attack never increases your Qi. Getting hurt by the bosses' special-skill attacks never increases your Qi, either.

As a perfectionist, you don't want any hero to be defeated even temporarily (in the real game you can rebirth a hero with certain Xian Shu or items) - for example, it's **not** allowed to make YTH and SY defeated at the same time (it can happen, for example, after YTH has performed a powerful physical special-skill attack).

Finally comes the question: what is the earliest time (we only care about ball-moving time, not including time needed to make attacks) that you can win the combat, if you play optimally? To play optimally, you need to choose between the options listed above, for each act. It's not as easy as it sounds. Be **careful**!

Input

There will be at most 100 test cases. Each case begins with 6 positive integers, *SY_jing* (SY's initial Jing), *XX_su* (XX's Su), *SY_su* (SY's Su), *yurun_jing*, *yurun_shen* and *shuerguo_shen*. The next four lines contain descriptions of YTH, HLS, LML and MRZY, in this order. Each line contains 16 non-negative integers: *maxjing*, *maxshen*, *su*, *d1x*, *d2x*, *d1s*, *d2s*, *wad*, *ssd*, *ssq*, *ssp*, *q*1, *q*2, *jing*, *qi*, *shen*. The last three parameters are the initial Jing, Qi and Shen values of this hero before the combat begins. The limits of most parameters are listed below:

Parameter(s)	min	max
SY_jing	1	100,000
yurun_jing, maxjing, d1x, d2x, d1s, d2s	1	8,000
yurun_shen, shuerguo_shen, maxshen	1	800
XX_su, SY_su, su	1	4
wad, ssd	1	100,000
ssp	0	1
*q*1, *q*2, *ssq*	1	100

And finally, 1 <= *jing* <= *maxjing*, 0 <= *qi* <= 100, 0 <= *shen* <= *maxshen*. The last test case is followed by 6 zeros, which should not be processed.

Output

For each test case, print the earliest time that you can win, and all possible orderings to achieve this. Each ordering is expressed as the concatenation of the first letters of H1, H2 and H3's names. For example, if H1=HLS, H2=LML, H3=YTH, the ordering is expressed as HLY. The orderings should be sorted in increasing order lexicographically. If you can't win the combat within 12 units of ball-moving time, print `-1`. Print a blank line after the output of each test case.

Sample Input

```
1000 1 1 200 15 75
1000 100 1 2000 2000 2000 2000 300 800 20 0 5 5
900 10 100
1000 100 1 2000 2000 2000 2000 120 300 10 0 5 5
100 80 100
1000 100 1 2000 2000 2000 2000 100 400 30 1 5 5
450 40 100
1000 100 1 2000 2000 2000 2000 250 700 10 1 5 5
600 50 100
3000 4 1 800 15 75
2000 100 3 2 2 2 2 1 1 1 0 2 1 1000 100 100
2000 100 4 2 2 2 2 1 1000 25 0 2 1 1 1 100
2000 100 1 2 2 2 2 1 1000 1 1 1 1 300 100 0
2000 100 3 2 2 2 2 1 1000 30 0 5 1 1 6 100
26399 3 2 3182 543 800
4462 353 2 4300 4875 6856 5527 31497 5633 61 0 68
63 4355 0 351
5444 300 3 7682 1037 597 4214 6744 6861 68 0 65 12
2136 32 143
5875 705 2 2097 118 2366 978 14276 24850 48 0 55
70 3562 40 277
6413 33 1 6305 1898 340 5238 13989 25287 25 1 72
34 3176 4 30
0 0 0 0 0 0
```

Output for the Sample Input

```
Case 1: 4 HLY HYL LHY LYH YHL YLH

Case 2: 12 HML

Case 3: -1
```

Explanation

In the first sample, you can win the combat just after one action for each hero (at time 4, since all the su values are 1), but you need to be careful.

YTH's special-skill attack is powerful (damage=900), but his Qi is not enough (10<20). You can't use HLS's weapon attack, since she'll get defeated by physical reflection. LML's weapon attack is weak, but she can use her special-skill attack even though she'll get hurt by reflection (her special-skill attack is physical), since 400<450. For MRZY, his special-skill is physical, and too powerful, so we can't use it, since 700>600.

To summarize, we can use YTH's weapon attack (damage=300), HLS's special-skill attack (damage=300), LML's weapon attack (damage=100) and special-skill attack (damage=400), and MRZY's weapon attack (damage=250). It's not hard to see, the only possible combination that can finish the combat in the quickest way is YTH, HLS and LML (use special-skill attack), the ordering is arbitrary. Note that the combat ends immediately after SY is defeated, so don't worry about the terrible attacks of XX and SY.

In the second sample, YTH is too weak so we ignore him immediately. You can't avoid XX's first attack, which is able to defeat everyone except LML, so LML seems to be the only possible H1. However, HLS moves before XX, so she can also be H1 if she uses Yu Run in her first action. Unfortunately, MRZY can't be H1, since he's not quick enough to recover Jing before XX attacks.

All the weapon attacks are too weak, so the best strategy is to accumulate Qi first, then perform the powerful special-skill attacks when possible. LML's su is too low, but her Qi is enough at startup. All she needs to do is to recover Shen, then recover Jing, and then do the attack (Why so complex? because her powerful special-skill attack is physical…). Note that SY is only able to attack two heroes before the combat ends, so H3 has one fewer chance to increase Qi by reflection. Actually, it can be proven that neither HLS nor MRZY can be H3 - if so, they'll be unable to gain enough Qi for their special-skill attacks.

To summarize, HLS can be H1 or H2, LML can be all, while MRZY can only be H1. So the only possible ordering is H1=HLS, H2=MRZY, H3=LML.

In the third sample, it seems that you can defeat SY with HLS's special-skill attack (damage=6861) and LML's special-skill attack (damage=24850). However, LML needs to recover Jing to avoid being defeated, but her initial Shen is not enough. If we changed her initial Shen to 543 (enough for Yu Run), the answer would become "6 LYH LYM".

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem G
Gift Hunting
Input: gift.in

After winning two coupons for the largest shopping mart in your city, you can't wait inviting your girlfriend for gift hunting. Having inspected hundreds of kinds of souvenirs, toys and cosmetics, you finally narrowed down the candidate list to only n gifts, numbered 1 to n. Each gift has a happiness value that measures how happy your girlfriend would be, if you get this gift for her. Some of them are special - you **must** get it for your girlfriend (note that whether a gift is special has nothing to do with its happiness value).

Coupon 1 can be used to buy gifts with total price not greater than $V1$ (RMB). Like most other coupons, you **can't** get any money back if the total price is strictly smaller than $V1$. Coupon 2 is almost the same, except that it's worth $V2$. Coupons should be used separately. That means you cannot combine them into a super-coupon that's worth $V1+V2$. You have to divide the gifts you choose into two part, one uses coupon 1, the other uses coupon 2.

It is your girlfriend's birthday today. According to the rules of the mart, she can take one (only one) gift for FREE! Here comes your challenge: how to make your girlfriend as happy as possible?

Input
There will be at most 20 test cases. Each case begins with 3 integers $V1$, $V2$ and n ($1 \le V1 \le 500$, $1 \le V2 \le 50$, $1 \le n \le 300$), the values of coupon 1 and coupon 2 respectively, and the number of candidate gifts. Each of the following n lines describes a gift with 3 integers: P, H and S, where P is the price, H is the happiness ($1 \le P,H \le 1000$), S=1 if and only if this is a special gift - you must buy it (or get it for free). Otherwise S=0. The last test case is followed by $V1=V2=n=0$, which should not be processed.

Output
For each test case, print the case number and the maximal total happiness of your girlfriend. If you can't finish the task, i.e. you are not able to buy all special gifts even with the 1-FREE bonus, the happiness is -1 (negative happiness means she's unhappy). Print a blank line after the output of each test case.

Sample Input	Output for the Sample Input
3 2 4 3 10 1 2 10 0 5 100 0 5 80 0 3 2 4 3 10 1 2 10 0 5 100 0 5 80 1 0 0 0	Case 1: 120 Case 2: 100

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem H
Help Bubu
Input: help.in

Bubu's bookshelf is in a mess! Help him!

There are *n* books on his bookshelf. We define the mess value to be the number of segments of **consecutive equal-height books**. For example, if the book heights are 30, 30, 31, 31, 32, the mess value is 3, that of 30, 32, 32, 31 is also 3, but the mess value of 31, 32, 31, 32, 31 is 5 - it's indeed in a mess!

Bubu wants to reduce the mess value as much as possible, but he's a little bit tired, so he decided to take out at most *k* books, then put them back somewhere in the shelf. Could you help him?

Input
There will be at most 20 test cases. Each case begins with two positive integers *n* and *k* ($1 \le k \le n \le 100$), the total number of books, and the maximum number of books to take out. The next line contains *n* integers, the heights of each book, from left to right. Each height is an integer between 25 and 32, inclusive. The last test case is followed by $n=k=0$, which should not be processed.

Output
For each test case, print the case number and the minimal final mess value. Print a blank line after the output of each test case.

Sample Input	Output for the Sample Input
5 2 25 25 32 32 25 5 1 25 26 25 26 25 0 0	Case 1: 2 Case 2: 3

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem I
In A Crazy City
Input: in.in

I live in a crazy city full of crossings and bidirectional roads connecting them. On most of the days, there will be a celebration in one of the crossings, that's why I call this city crazy.

Everyday, I walk from my home (at crossing *s*) to my office (at crossing *t*). I don't like crowds, but I don't want to waste time either, so I always choose a shortest path among all possible paths that does not visit the crossing of the celebration. If no such path exists, I don't go to work (it's a good excuse, isn't it)!

In order to analyze this "celebration effect" in detail, I need *n* pairs of values (l_i, c_i), where l_i is the length of the shortest path from crossing *s* to crossing *t*, not visiting crossing *i*, c_i is the number of such shortest paths (not visiting crossing *i*). Could you help me? Note that if I can't go to work when celebration is held at crossing *i*, define $l_i=c_i=0$. This includes the case when there is no path between *s* and *t* even if there's no celebration at all.

Ah, wait a moment. Please don't directly give me the values - that'll drive me crazy (too many numbers!). All I need is finding some interesting conclusions behind the values, but currently I've no idea what exactly I want.

Before I know what you should calculate, please **prove** that you can indeed find all the pairs (l_i, c_i) by telling me the value of $f(x) = (l_1+c_1x+l_2x^2+c_2x^3+l_3x^4+c_3x^5+\ldots+l_nx^{2n-2}+c_nx^{2n-1}) \bmod 19880830$, for some given *x*.

Input
There will be at most 20 test cases. Each case begins with 5 integers *n*, *m*, *s*, *t*, *q* ($1\le s,t\le n\le 100{,}000$, $0\le m\le 500{,}000$, $1\le q\le 5$). *n* is the number of crossings, *m* is the number of roads and *q* is the number of queries. *s* and *t* are different integers that represent my home and office, respectively. Each of the following *m* lines describes a road with three integers: *u*, *v*, *w* ($1\le u,v\le n$, $1\le w\le 10{,}000$), indicating a bidirectional road connecting crossing *u* and crossing *v*, with length *w*. There may be multiple roads connecting the same pair of crossings, but a road cannot be connecting a crossing and itself. The next line contains *q* integers x_i ($1\le x_i\le 10^9$). The last test case is following by five zeros, which should not be processed.

Output
For each test case, print the case number and *q* integers $f(x_1)$, $f(x_2)$, …, $f(x_q)$ separated by a single space between consecutive items, on one line. Print a blank line after the output of each test case.

Sample Input	Output for the Sample Input
4 5 1 4 2 1 2 1 1 3 1 2 4 2 3 4 3 1 4 4 1 10 3 2 1 3 1 1 2 12 2 3 2 1 0 0 0 0 0	Case 1: 10 132400 Case 2: 0

Explanation
In the first sample, $l_1=c_1=0$, $l_2=4$, $c_2=2$, $l_3=3$, $c_3=1$, $l_4=c_4=0$. In the second sample, everything is zero.

The 2009 ACM Asia Programming Contest Wuhan Site
sponsored by IBM
hosted by Wuhan University

Problem J

Jiajia's Robot

Input: jiajia.in

Jiajia wants to do some experiments on his two-eyed robot. Each eye of the robot shoots a very thin laser beam from it, which goes infinitely (i.e. a mathematical "ray"). Restricted by the internal structure, the two laser beams from the eyes must always form a right angle (i.e. two rays must be perpendicular to each other).

To help the robot localize itself, Jiajia placed two linear-shaped special materials called MA and MB. If one of the rays intersects with MA, while the other ray intersects with MB, the robot is able to gather enough spatial information for the localization.

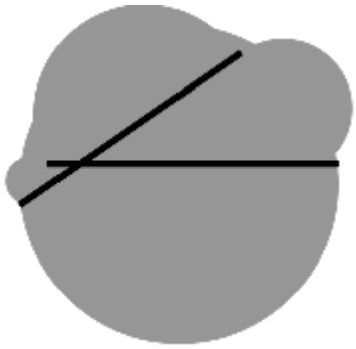

Fig. The collection of points from which the robot can localize itself.

The weird shape in the picture above is the collection of points from which the robot can localize itself. MA and MB are drawn as two line segments.

Though the task seems a little bit difficult, can you tell me the total area of these points?

Input

There will be at most 50 test cases. Each case contains a single line of 8 positive integers x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 not greater than 500, where (x_1,y_1) and (x_2,y_2) are two endpoints of MA, (x_3,y_3) and (x_4,y_4) are two endpoints of MB. Note that MA and MB can have at most one common point. Neither MA nor MB will be degenerated to a point. The last test case is followed by 8 zeros, which should not be processed.

Output

For each test case, print the total area of "self-localizable point", to 3 digits after the decimal point. Print a blank line after the output of each test case.

Sample Input	Output for the Sample Input
264 280 147 360 162 335 320 334 203 165 288 227 149 295 153 344 0 0 0 0 0 0 0 0	Case 1: 26634.633 Case 2: 27436.383