A. On the Run

The world's funniest schoolteacher has found himself in some trouble. Or, should we say, the world's funniest ex-schoolteacher. On the basis of a false accusation regarding too many puns in his exam problems, Mr. X has not only been fired, but is now on the run from the Humane Association for Humour Administration (HAHA).

They've tracked Mr. X down to a yard in Scotland, which may be represented as a grid with N rows (numbered 1 to N) and M columns (numbered 1 to M). Mr. X is initially in row A and column B. There are also K ($1 \le K \le 2$) HAHA agents hot on his trail, the *i*th of whom is initially in row R_i and column C_i. All K+1 individuals are in distinct cells.

The chase will then commence in an organized fashion as follows:

1. Each of the K HAHA agents in turn will move up, down, left, or right to an adjacent unoccupied cell* (without leaving the yard). The agents may choose to move in any order, but each of them must move exactly once, and multiple of them may not move simultaneously. It's guaranteed that all of the agents will always be able to move in some order for any possible state of the yard.
2. Mr. X will then attempt to similarly move to an adjacent unoccupied cell. If he's unable to move due to there being no unoccupied cells adjacent to him, he'll surrender quietly instead. Otherwise, the process will repeat from Step 1.

* *An unoccupied cell is one which currently contains neither Mr. X nor an agent.*

Mr. X is hoping that, if he can avoid ever being forced to surrender, the HAHA agents may eventually leave him alone, giving him the opportunity to slip away and work on regaining his teaching position. However, the outcome of this chase seems difficult to call. Assuming that the agents work together optimally in an attempt to force Mr. X to surrender, while Mr. X optimally chooses moves to avoid surrendering indefinitely, determine whether or not he will eventually be forced to surrender.

Input

Input begins with an integer T, the number of times that Mr. X is chased by HAHA agents. For each chase, there is first a line containing the space-separated integers N, M, and K. Then there is a line containing the space-separated integers A, and B. Then, K lines follow, the *i*th of which contains the space-separated integers R_i and C_i.

Output

For the *i*th chase, print a line containing "Case #*i*: " followed by one character, either "Y" if Mr. X will eventually be forced to surrender, or "N" otherwise.

Constraints

$1 \le T \le 500$

$3 \le N, M \le 300$

$1 \le K \le 2$

$1 \le A, R_i \le N$

$1 \le B, C_i \le M$

Explanation of Sample

In the first case, no matter how much time goes by and what moves the HAHA agent chooses to make, Mr. X can always avoid needing to surrender.

In the second case, if the first HAHA agent initially moves to cell (1, 2) and the second agent moves to cell (2, 1), then Mr. X will immediately be forced to surrender.

Example Input

```
8
3 3 1
1 1
2 2
3 3 2
1 1
1 3
3 1
4 4 2
1 2
1 3
1 4
3 10 2
2 10
1 9
3 9
8 8 2
8 1
8 8
1 1
300 5 2
2 3
15 2
300 5
67 25 2
32 10
66 3
21 18
```

```
71 87 2
36 44
1 87
71 1
```

Example Output

```
Case #1: N
Case #2: Y
Case #3: N
Case #4: Y
Case #5: N
Case #6: Y
Case #7: N
Case #8: Y
```

B. Bitstrings as a Service

Carlos knows that the key to sustained customer loyalty is early customer acquisition and constant re-engagement. With that in mind, he's setting his sights on sugary cereals. Kids love it, and they keep coming back for more. He has a proposal for Post, the well-known manufacturer of Alpha-Bits:

"For 60 years these bits have been made of letters, but kids today are tired of letters for breakfast. Let me show you something truly revolutionary: bits made of numbers!"

The executives at Post would like to assess the feasibility of producing numeric bits. In particular, they want to see if Carlos can create a string of bits, B, of length N. In this string, each character is either `'0'` or `'1'`.

Carlos has been given a list of M requirements. The *i*th requirement states that the substring $B_{X_i...Y_i}$ must be a palindrome. The new cereal surely won't succeed without appealing to children's acute sense of symmetry.

Furthermore, the number of zeroes and ones in B must be as close as possible. In other words, if `C(b)` is the number of occurrences of bit *b* in B, then the absolute difference `|C(0) - C(1)|` must be minimized. The margins on cereal are razor thin, and as everybody knows, bits are produced in pairs, so it's wasteful to use more of one sort than the other.

Help Carlos build any bitstring B consistent with all of these requirements (and with minimal difference between the number of zeroes and ones). It's guaranteed that such a bitstring exists for every given set of requirements.

Input

Input begins with an integer T, the number of bitstrings that Post has commissioned. For each bitstring, there is first a line containing the space-separated integers N and M. Then, M lines follow, the *i*th of which contains the space-separated integers X_i and Y_i.

Output

For the *i*th commission, print a line containing "Case #*i*: " followed by a valid bitstring B.

Constraints

$1 \le T \le 350$

$1 \le N \le 4{,}000$

$0 \le M \le 10{,}000$

$1 \le X_i \le Y_i \le N$

Explanation of Sample

In the first case, the minimum absolute difference between the number of zeroes and ones is 0 (with 2 of each). 5 other outputs would also be accepted for this case: "0011", "0101", "0110", "1001", and "1010". For each of the remaining sample cases, various other outputs would similarly be accepted.

In the second case, the minimum achievable difference is 2.

In the third case, the minimum achievable difference is 2.

In the fourth case, the minimum achievable difference is 1.

In the fifth case, the minimum achievable difference is 0.

In the sixth case, the minimum achievable difference is 9.

Example Input

```
6
4 0
6 1
1 6
4 2
1 2
2 4
5 3
3 5
2 2
2 4
10 5
3 6
1 4
6 8
5 9
9 10
25 10
17 20
```

```
11 15
2 3
20 22
8 8
24 25
7 12
1 8
14 18
15 21
```

Example Output

```
Case #1: 1100
Case #2: 110011
Case #3: 1101
Case #4: 01010
Case #5: 1001100011
Case #6: 000110001100100000000111
```

C. Grading

Due to a convenient recent teaching vacancy, Laz Y. has suddenly landed a job as a schoolteacher. Known to his students as Mr. Y, he's prepared to provide a comprehensive, fairly-evaluated educational experience — as long as it doesn't take too much effort.

Mr. Y's first order of business in his new role will be grading recent exams from two different subjects: art and biology. He has been handed S stacks of H exam papers each. The *i*th paper from the top in the *j*th stack is either from the art exam (if $P_{i,j}$ = "A"), or otherwise from the biology exam (if $P_{i,j}$ = "B").

Mr. Y will go about the grading process as follows: At each point in time, he'll select a stack which still contains at least one exam paper, and remove its topmost paper. He'll then either grade that paper, or accidentally "lose" it and assign its owner a random grade instead. Either way, once he's done with that paper, he'll repeat the process of selecting a new paper until all of the stacks are empty and all H*S papers have been dealt with.

There are few things that Mr. Y hates as much as context switching. For example, it's very troublesome to jump from grading an art exam to a biology one! (Or from relaxing to doing any work at all.) Each time Mr. Y begins a grading a paper, he must make a context switch if this is either the first paper he's choosing to grade, or if its subject is different than that of the previous paper that he graded. Note that this entirely excludes any "lost" papers.

Mr. Y is no fool — he realizes that his evaluations would be too suspiciously inaccurate if he were to simply lose all H*S papers. Even losing a smaller number of them may prove too suspicious. Therefore, he'll imagine K different theoretical scenarios, such that in the *i*th one, he will allow himself to lose at most L_i papers throughout the grading process (with $L_{1..K}$ all being distinct). Independently for each scenario, he'd like to determine the minimum number of context switches he would need to make throughout the process.

Input

Input begins with an integer T, the number of days Mr. Y spends grading exams. For each day, there is first a line containing the space-separated integers H, S, and K. Then, H lines follow, the *i*th of which contains the length-S string $P_{i,1..S}$. Then, a final line follows containing the K space-separated integers L_1 through L_K.

Output

For the *i*th day, print a line containing "Case #*i*: " followed by K space-separated integers, the *j*th of which is the minimum number of context switches which Mr. Y would need to make if he were to grade the papers while losing at most L_j of them.

Constraints

$1 \le T \le 200$

$1 \le H, S \le 300$

$1 \le K \le H * S$

$0 \le L_i \le H * S - 1$

Explanation of Sample

In the first case, if Mr. Y may only lose one paper, the best he can do is make two context switches, for example by grading the B papers from the first and third stacks, then losing the B paper from the fifth stack, and then grading the A papers from the second and fourth stacks. On the other hand, the freedom to lose two papers would allow him to make only one context switch, for example by losing the A papers from the second and fourth stacks and then grading the B papers from the first, third, and fifth stacks.

In the second case, if Mr. Y may lose one paper, one optimal strategy (requiring just two context switches) is as follows:

1. Grade the B paper from the top of stack 2 (first context switch).
2. Lose the A paper from the top of stack 3.
3. Grade the B paper from the top of stack 3.
4. Grade the A paper from the top of stack 1 (second context switch).
5. Grade the A paper from the top of stack 1.
6. Grade the A paper from the top of stack 2.

Example Input

```
5
1 5 2
BABAB
1 2
2 3 3
ABA
AAB
1 0 5
3 2 4
AB
BA
AB
0 1 2 3
5 5 6
BBABA
ABAAB
AAABA
BABBA
BBBAB
5 0 8 12 10 1
10 10 15
AABAAABBAB
BAABAAAABB
AABABBBABB
BAAABAAAAB
BBBBAAABAA
ABAABBBABA
BABAABABBA
AAABAAABAA
BAAAABBBBA
ABABBAAABA
14 2 99 33 3 8 43 4 12 1 21 24 17 32 10
```

Example Output

```
Case #1: 2 1
Case #2: 2 3 1
Case #3: 4 3 2 1
Case #4: 3 5 2 1 2 4
Case #5: 5 8 1 2 8 7 1 7 5 9 4 3 4 3 6
```

D. Seafood

Percy is a harlequin tuskfish who lives on a stretch of the sea floor, which may be represented as a number line. There are N objects resting on the sand beneath the waves, the *i*th of which is at a positive integral position P_i, is either a clam (if O_i = "C") or otherwise a rock (if O_i = "R"), and in either case has a hardness of H_i. No two objects are at the same position, and at least one of the objects is a clam.

Percy initially finds himself at position 0, and can then swim in either direction along the number line at a rate of 1 unit per second. Whenever he occupies the same position as a clam, he may pick it up and begin carrying it around in his mouth. Picking up a clam requires no additional time, and Percy is talented enough to carry any number of clams at once.

Percy would like to devour the delicious interior of each clam, but can't get to it without first somehow breaking open its hard shell. Fortunately, Percy is clever and persistent enough to have a solution to this problem. Whenever he occupies the same position as a rock, he may take each clam that he's currently carrying that has a strictly smaller hardness than that of the rock, knock the clam against the rock to break open its shell, and eat the meal packaged within! This process requires no additional time per clam.

What's the minimum amount of time required for Percy to swim around and eat all of the clams (by picking each one up and then knocking it against a harder rock than itself), if at all possible?

In order to reduce the size of the input, the object's positions and hardnesses will not all be provided explicitly. Instead, you'll be given P_1, P_2, H_1, H_2, as well as the 8 constants A_p, B_p, C_p, D_p, A_h, B_h, C_h, and D_h, and must then compute $P_{3..N}$ and $H_{3..N}$ as follows (bearing in mind that intermediate values may not fit within 32-bit integers):

$P_i = ((A_p * P_{i-2} + B_p * P_{i-1} + C_p)$ modulo $D_p) + 1$, for $i = 3$ to N.

$H_i = ((A_h * H_{i-2} + B_h * H_{i-1} + C_h)$ modulo $D_h) + 1$, for $i = 3$ to N.

Input

Input begins with an integer T, the number of days Percy goes hunting for clams. For each day, there are four lines. The first line contains the integer N. The second line contains the space-separated integers P_1, P_2, A_p, B_p, C_p, and D_p. The third line contains the space-separated integers H_1, H_2, A_h, B_h, C_h, and D_h. The fourth line contains the length-N string $O_{1..N}$.

Output

For the *i*th day, print a line containing "Case #*i*: " followed by one integer, either the minimum number of seconds required for Percy to break open and eat all of the clams, or -1 if he cannot do so.

Constraints

$1 \le T \le 250$

$2 \le N \le 800{,}000$

$0 \le A_p, B_p, C_p, A_h, B_h, C_h \le 1{,}000{,}000{,}000$

$1 \le D_p, D_h \le 1{,}000{,}000{,}000$

$1 \le P_i \le D_p$

$1 \le H_i \le D_h$

Explanation of Sample

In the first case, P = [5, 10] and H = [30, 31]. Percy should swim to position 5, pick up the clam with hardness 30, swim onwards to position 10, and break open the clam on the rock with hardness 31.

In the second case, P = [5, 10] and H = [31, 30]. Percy should now swim to position 10 to pick up the clam, and then back to position 5 to break it open.

In the third case, P = [5, 10] and H = [30, 30]. Once Percy picks up the clam, no rock harder than it exists on the sea floor, meaning that he can never break it open.

In the fourth case, P = [10, 50, 11, 52] and H = [50, 10, 49, 8].

In the fifth case, P = [415, 711, 225, 136, 256, 469, 714, 399, 841, 697, 480, 147, 98, 837, 745, 660, 44, 226, 73, 7] and H = [9, 2, 10, 6, 6, 2, 7, 13, 4, 5, 11, 11, 4, 3, 7, 1, 6, 10, 10, 1].

Example Input (long lines are wrapped)

```
6
2
5 10 0 0 0 50
30 31 0 0 0 50
CR
2
5 10 0 0 0 50
31 30 0 0 0 50
RC
2
5 10 0 0 0 50
30 30 0 0 0 50
RC
4
10 50 0 1 40 80
50 10 0 1 38 80
RRCC
20
415 711 3 4 3 967
9 2 1 2 9 13
CCCCCRRRCRCCCRRCRRCC
100
168981242 670860915 208968638 604295408 490937286
715757945
627165633 146096256 952913201 456337362 978266551
970054933
CRCCRRRRRRRRCRRCCCRCCRRRRCRCRRRRCCCCRCCRRRCRRRCCC
CCRCCRCRCCCCCRRCCCRRCRRRRCCCRCRRCRRRRCCCRRCRCCCCC
RR
```

Example Output

```
Case #1: 10
Case #2: 15
Case #3: -1
Case #4: 56
Case #5: 1099
Case #6: 890508817
```