Problem A
Totally Monotone Matrix
Max no. of test cases: 15
Time limit: 1 second

An nxn matrix M is called *totally monotone* if elements in each row and each column are all non-decreasing. In this problem, all elements of matrix M have distinct values. The example below shows a 5x5 totally monotone matrix, where $M[1,1] = 2$, $M[1,5] = 19$ and $M[5,5] = 72$.

```
 2  5 10 13 19
 3  7 11 15 22
 4  9 16 20 43
 8 14 17 31 66
12 21 26 40 72
```

Give a totally monotone matrix M and two elements at $M[r_1, c_1]$ and $M[r_2, c_2]$, where $1 \leq r_1 < r_2 \leq n$ and $1 \leq c_1 < c_2 \leq n$. Please determine the number of elements in M that has value greater than $M[r_1, c_1]$ and smaller than $M[r_2, c_2]$. In the above totally monotone matrix M, there are 11 elements with values greater than $M[2, 2]$ (which is 7) and less than $M[3, 4]$ (which is 20), namely 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19.

Input File Format

First line of input has one integer, indicating the number of test cases. For each test case, the first line contains an integer n, $3 \leq n \leq 1000$, which is the size of the totally monotone matrix M to follow. The next n lines each has n integers, representing the n rows of n columns of M. The last line contains 4 integers, r_1, c_1, r_2, and c_2, indicating the indices of elements $M[r_1, c_1]$ and $M[r_2, c_2]$.

Output Format

For each test case, output an integer on a single line, indicating the number of elements in M that are larger than $M[r_1, c_1]$ and smaller than $M[r_2, c_2]$.

Sample Input

```
2
5
2 5 10 13 19
3 7 11 15 22
4 9 16 20 43
8 14 17 31 66
12 21 26 40 72
2 2 3 4
3
1 2 5
3 7 13
9 11 15
1 1 1 2
```

Output for the Sample Input

```
11
0
```

Problem B
Aircraft Tracking Radar
Max no. of test cases: 15
Time limit: 1 second

Flight Radar Systems are critical to the detection and tracking of aircrafts. The systems ensure the air traffic controllers know where the aircrafts are at every moment of flights. It is important for the controllers to identify each flight on the radar screen and to ensure aircrafts keep a safe distance from each other.

In one complete radar scan, information such as aircraft ID, altitude, destination, ground speed, etc., are shown on the radar screen for controllers to see. For this problem, let's assume the aircraft IDs are unique 5 digit positive integers, and the aircraft location in the sky is transformed into (ID, x, y, alt), where ID is the unique aircraft ID, x, y are coordinates on the radar screen, and alt is altitude of the aircraft. The system uses only integer values.

For the next system upgrade, aircraft proximity warning is to be activated to help aircraft controllers monitor those aircrafts. Two aircrafts, (ID$_i$, x_i, y_i, alt_i) and (ID$_j$, x_j, y_j, alt_j) are too close and thus need special monitoring if

1. the roundup Euclidean distance on the radar screen is less than or equal to a preset distance threshold, D_{xy}, or
2. the roundup Euclidean distance on the radar screen is less than or equal to 2xD_{xy} AND altitude difference is less than or equal to D_{alt}.

Please write a program to find all pair of aircrafts that need special monitoring by the aircraft controllers.

Input File Format

First line of input has one integer, indicating the number of test cases. For each test case, the first line contains three integers n, D_{xy}, and D_{alt}, which are the number aircrafts ($1 \leq n \leq 5000$), the Euclidean distance threshold and the altitude threshold, respectively. For the next n lines, each line contains 4 integers to represent one aircraft, namely ID, x, y, alt, where $10000 \leq$ID≤ 99999, $-10^5 \leq x, y \leq 10^5$, and $10000 \leq alt \leq 50000$, respectively.

Output Format

For each test case, output all aircrafts that requires special monitoring by the aircraft controllers according to the conditions set forth in the problem statement. Each line should contain 4 integers, ID_i, ID_j, E_{dist}, and Alt, where ID_i and ID_j ($\mathrm{ID}_i < \mathrm{ID}_j$) are aircraft IDs, E_{dist} is the roundup Euclidean distance on the radar screen of these two aircrafts, and alt is the difference in altitude between these two aircrafts. All output should be ordered by aircraft IDs from smaller ID to larger ID. The last line for each case should be an integer indicating the total number of warnings generated.

Sample Input

```
2
5 5000 1000
50000 -1400 1500 9500
30000 120 1200 9535
20000 120 1200 10000
40000 -1000 1300 10000
10000 110 1500 8500
3 10000 2000
22222 -100 100 8900
11111 -100 200 8900
55555 100 300  12000
```

Output for the Sample Input

```
10000 20000 301 1500
10000 30000 301 1035
10000 40000 1128 1500
10000 50000 1510 1000
20000 30000 0 465
20000 40000 1125 0
20000 50000 1550 500
30000 40000 1125 465
30000 50000 1550 35
40000 50000 448 500
10
11111 22222 100 0
11111 55555 224 3100
22222 55555 283 3100
3
```

Problem C
Breaking a Magnetic Plate
Max no. of test cases: 6
Time limit: 1 second

A scientist created a magnetic plate. He divided the plate into four squares and measured the magnetic field at the center of the squares. Fig. 1 shows an example of such a plate and its square magnetic values. When he wanted to measure the total magnetic field for the plate in Fig.1, he got 12. After days in research, he found the formula to compute the total magnetic field value. The secret is that squares can interact with each other. For example, square (0,0) (value 3) has neighbors (0,1) (value 2) and (1,0) (value 1). So, the "enhanced" magnetic value of square (0,0) is $3 * 2 + 3 * 1 = 9$, that is, by multiplying its magnetic value with its neighbor's value and then summing up all the neighbors. Diagonal connected squares, such as (0,0) and (1,1) are not neighbors.

By computing the enhanced values of every square and then adding all the values, you can get the total magnetic value of the plate. For example, the "enhanced" value of square (0,0) is 9, (0,1) is 4, (1,0) is 2, and (1,1) is -3, so the total magnetic value of the plate is 12.

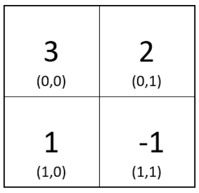

Figure 1: A magnetic plate

The scientist later found that by breaking the plate, sometimes, you can get a sub component to emit higher magnetic fields. For example, in Fig. 2, the piece composed by (0,0),(0,1),(1,0) can emit magnetic field value to 18 and (1,1) has value -1. The total magnetic field value is 17.

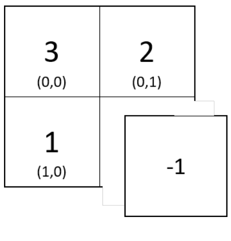

Figure 2: Break a magnetic plate

Given a plate, please find a way to break the plate so that the total magnetic field value is maximum.

Input File Format

There are more than one test cases in the input file. The first number N is the number of test cases. Each test case contains 4 integers separated by spaces. The 4 integers (0 is a feasible value) are the magnetic values of (0,0), (0,1), (1,0), and (1,1) respectively.

Output Format

For each test case, please print its maximum total magnetic value.

Sample Input

```
4
3 2 1 -1
1 1 1 1
3 2 -1 -1
3 2 -6 0
```

Output for the Sample Input

```
17
8
14
12
```

Problem D
Extend Huffman Coding
Max no. of test cases: 10
Time limit: 2 seconds

Huffman coding is a well-known data compression method. Assume a source generates 4 different symbols $\{A, B, C, D\}$ with probability $\{0.4; 0.35; 0.2; 0.05\}$. A binary tree is generated from left to right (sorting by alphabetic order of symbols) taking the two least probable symbols and putting them together to form an internal node which has a probability that equals the sum of the two symbols. When there are equal probability nodes, the selection of the two least probable nodes always follow the alphabetic order. For example, if a node is formed by $\{B, C\}$ and a node is formed by $\{A, D\}$, node $\{A, D\}$ is at the left of node $\{B, C\}$ because $A < B$ and $A < C$, as in the alphabetic order.

The process is repeated until there is just one node with probability value of 1.

The tree can then be read downwards, from left to right, assigning different bits to different branches. The tree construction is illustrated in Fig. 1.

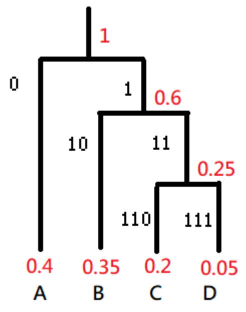

Figure 1: Huffman Binary Tree Construction

The final Huffman code is:

Symbol	*code*
A	0
B	10
C	110
D	111

So, "AAABC" will be encoded as 00010110.

The Huffman coding, however, can be extended by considering more symbol combinations. For example, symbol A's probability is 0.4, then AA's probabiliy is $0.4 * 0.4 = 0.16$. Symbol B's probability is 0.35 then the probability of AB is $0.4 * 0.35 = 0.14$. Let's call it a 2-symbol probability.

In Fig. 2, we show the beginning of how such a 2-symbol tree should be constructed based on the aforementioned alphabetic ordering. In this example, symbol 'CD' and 'DC' have equal probability. By the alphabetic ordering, 'CD' should be selected to merge with 'DD'.

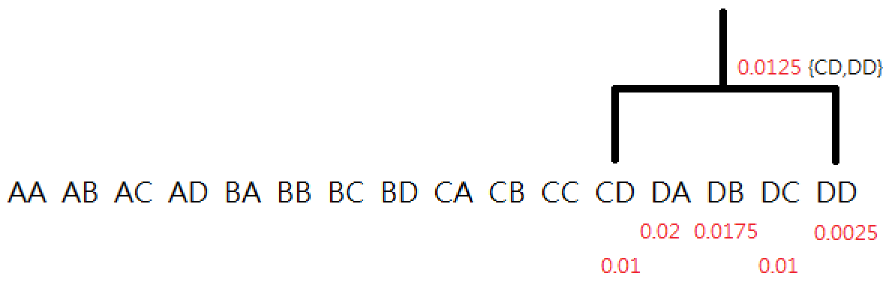

Figure 2: Huffman Binary Tree Construction

Given a set of symbols and their 1-symbol probabilities, please construct 2-symbol Huffman coding to encode a string. sorting by alphabetic order of symbols

Input File Format

There are more than one test cases in the input file. The first number N is the number of test cases. Each test case begins with $n(n <= 10)$ which is the number of symbols. It is followed by n lines of symbol (single character from [A-Z] or [a-z]) and its 1-symbol probability. The last line of the test case is a symbol string with length <= 36, which is an even number.

Output Format

For each test case, please print the length (int bits) of the encoded 0/1 string.

Sample Input

```
2
4
A 0.6
B 0.2
C 0.1
D 0.1
AAABDD
3
a 0.45
b 0.35
c 0.20
baac
```

Output for the Sample Input

```
11
7
```

Problem E
Ancient Tablet
Max no. of test cases: 45
Time limit: 5 seconds

In year 2030, the National Cryptography Programming Challenge (NCPC) Museum has an ancient digital tablet on display and for interactive playing. When enter a positive integer x, the tablet shows a sequence of numbers. JP visited the museum and played with the tablet with the following results.

JP entered number 3, the tablet showed "2 3 1 2 1 3".
JP entered number 4, the tablet showed "2 3 4 2 1 3 1 4".
JP entered number 5, the tablet showed "0".

After playing a few more rounds, JP soon discovered the secret behind this tablet. For a given number x, if possible, a sequence of numbers is displayed. The sequence is made up of numbers between 1 and x. Each number appeared exactly twice. Furthermore, there is

- exactly one number between the two "1s" in the sequence,
- exactly two numbers between the two "2s" in the sequence,
- exactly three numbers between the two "3s" in the sequence,
- . . .
- exactly x numbers between the two x in the sequence.

JP also noted that if no such sequence exists, then an "0" is displayed.

Please write program to mimic the behavior of this old tablet.

Input File Format

The first line of input contains an integer n, indicating the number of test cases. For each test case, there is exactly one positive integer x, $1 \leq x \leq 45$, on a single line.

Output Format

For each test case, output either a single "0", or a sequence of $2x$ integers that meet the description above on a single line. Integers on the same line should be separated by blank space.

Sample Input

```
4
3
4
5
8
```

Output for the Sample Input

```
2 3 1 2 1 3
2 3 4 2 1 3 1 4
0
1 5 1 4 6 7 8 5 4 2 3 6 2 7 3 8
```

Problem F
Sequence of Numbers
Max no. of test cases: 25
Time limit: 1 second

A **look-and-say sequence** is shown as follows:

$1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, 31131211131221, \ldots$

To generate a member of the sequence from the previous member, one would repeat the following process until end of that number: count and say the number of consecutive same digit followed by saying that digit. For example:

- 1 is read off as "one 1", represented by 11.
- 11 is read off as "two 1s" represented by 21.
- 21 is read off as "one 2, one 1" represented by 1211.
- 1211 is read off as "one 1, one 2, two 1s" represented by 111221.
- 111221 is read off as "three 1s, two 2s, one 1" represented by 312211.
- . . .

Now, we consider a variation of the look-and-say sequence as follows:

Given an arbitrary integer x, a number in the sequence is generated from the previous number by *counting the number of occurrences of digits* 0-9 *in order* and reading it off. A digit with zero occurrence should not be read off.

For example, given the integer 1, the generated sequence is as follows: "1, 11, 21, 1112, 3112, 211213, 312213, 212223, 114213, 31121314, 41122314, 31221324, 21322314, 21322314, 21322314, . . . ".

- 1 is read off as "one 1", represented by 11.
- 11 is read off as "two 1s" represented by 21.
- 21 is read off as "one 1, one 2" represented by 1112.
- 1112 is read off as "three 1s, one 2" represented by 3112.
- 3112 is read off as "two 1s, one 2, one 3" represented by 211213.
- . . .
- 31221324 is read off as "two 1s, three 2s, two 3s, one 4" represented by 21322314.
- 21322314 is read off as "two 1s, three 2s, two 3s, one 4" represented by 21322314.

- 21322314 is read off as “two 1s, three 2s, two 3s, one 4” represented by 21322314.
- . . .

This sequence number begins to repeat starting at the 13th number.

As another example, given integer 11111111111, the sequence is “11111111111, 111, 31, 1113, 3113, 2123, 112213, 312213, 212223, 114213, 31121314, 41122314, 31221324, 21322314, 21322314, 21322314, . . . ”.

- 11111111111 is read off as “eleven 1s”, represented by 111.
- 111 is read off as “three 1s” represented by 31.
- 31 is read off as “one 1, one 3” represented by 1113.
- 1113 is read off as “three 1s, one 3” represented by 3113.
- 3113 is read off as “two 1s, two 3s” represented by 2123.
- . . .
- 31221324 is read off as “two 1s, three 2s, two 3s, one 4” represented by 21322314.
- 21322314 is read off as “two 1s, three 2s, two 3s, one 4” represented by 21322314.
- 21322314 is read off as “two 1s, three 2s, two 3s, one 4” represented by 21322314.
- . . .

And the sequence number begins to repeat starting at the 14th number.

Please write a program to take a starting number x and determine where in the sequence does the number starts to repeat.

Input File Format

First line of the input contains one integer indicating the number of test cases. For each test case, there is one positive integer x, $1 \leq x \leq 10^{20}$, on a single line.

Output Format

For each test case, output an integer on a single line, indicating where in the generated sequence does the number starting to repeat.

Sample Input

```
4
1
22
126
11111111111
```

Output for the Sample Input

```
13
1
10
14
```

Problem G
Website Hits
Max no. of test cases: 16
Time limit: 4 seconds

The NCPC practice website is the goto site for programming practices. The website keeps a count of number of people visiting the website each day. After many months in service, there are now n days of website hit count, namely $h_1, h_2, ..., h_n$, available for analysis. The website manager wants to know if there are trends in the hit counts, so he decides to do the following analysis.

Define the k-day hit count interval starting at day i to be $h_i, h_{i+1}, h_{i+2}, ..., h_{i+k-1}$. Thus, there can be exactly $n-k+1$ k-day intervals, starting from day $1, 2, ...$, and day $n-k+1$, respectively. A pair of k-day intervals is called m-similar if the number of corresponding days that have different hit count is at most m. In other words, if $k_i = [h_i, h_{i+1}, \ldots, h_{i+k-1}]$ and $k_j = [h_j, h_{j+1}, \ldots, h_{j+k-1}]$, k_i and k_j are m-similar if $\sum_{c=0}^{k-1}(h_{i+c} \neq h_{j+c}) \leq m$.

For example, if the hit counts collected are $34, 56, 96, 78, 34, 56, 60, 78, 52, 96, 60, 30$. then 7-day interval ($k = 7$) starting at day 1 and day 5 are $k_1 = 34, 56, 96, 78, 34, 56, 60$ and $k_5 = 34, 56, 60, 78, 52, 96, 60$. The two 7-day intervals have 3 corresponding days with different hit counts (day 3, 5, 6). Thus intervals k_1 and k_5 are at least 3$-$similar (can also be said to be 4$-$, 5$-$, 6$-$, or 7$-$similar). For this example, total number of pair of intervals that is 4-similar is 2 (k_1–k_5 and k_2– k_6).

Given the hit counts and the interval length k, please help determine the total number of pair of intervals that are $m-$similar in a sequence of queries.

Input File Format

The first line of input contains an integer indicating the number of test cases. For each test case, the first line contains three integer, n, k, and q which are the number of consecutive days of hit count collected, the interval length to be analyzed, and the number of $m-$similar interval pairs to be computed. Note that $1 \leq k \leq n \leq 10,000$ and $q \leq 100$. The next line contains n integers, indicating the number of hit counts in n consecutive days. Note that the hit counts are all positive integers but no greater than 10^9. The next q lines each contains an integer m, which is a query asking for the number of $m-$similar k-day interval pairs in the given list of hit counts.

Output Format

For each test case, output q integers on a single line, which are the total number of $m-$similar interval pairs among all k-day intervals for each of the q queries in the input. The answer should be in sequence of the q queries.

Sample Input

```
2
12 7 3
34 56 96 96 34 56 60 96 52 96 60 30   <-- 12 hit counts ==> six 7-day interval
4                   <-- query on number of 4-similar 7-day interval pairs
6                   <-- query on number of 6-similar 7-day interval pairs
2                   <-- query on number of 2-similar 7-day interval pairs
4 2 2
1 2 1 4             <-- four hit counts ==> three 2-day interval
1                   <-- query on number of 1-similar 2-day interval pairs
2                   <-- query on number of 2-similar 2-day interval pairs
```

Output for the Sample Input

```
2 9 0
1 3
```