Fundamentals of Algorithms

Grade of Computer Science. Group I

Examination Ordinary Call, January 21, 2022.

Rules for conducting the exam

1. You have to program solutions for each of the three exercises, test them and submit them to the automatic judge accessible at the address `http://exacrc/domjudge/team`. For the submission evaluation, the judge has only the test data from the problem statement.
2. Write comments that explain your solution, justify why it has been done this way and help to understand it. Calculate the complexity of all the functions you implement.
3. At the judge you will identify yourself with the username and password you received at the beginning of the exam. The username and password you have been using during the continuous evaluation are **not** valid.
4. Write your **first and last name** in a comment on the first line of each file you upload to the judge.
5. Your solutions will be evaluated by the teacher independently of the verdict of the automatic judge. For this purpose, the teacher will **only** take into account the last submission you have made for each exercise.

1. **(3.5 points)** Given a vector v of $n \geq 0$ positive integers, we want to count the number of non-empty segments that satisfy that all their elements are even.

    1. (0.25 points) (Specification, NO implementation) Define a predicate $allEven(v, p, q)$ that returns true if and only if all the elements of the vector v contained between the positions p (incluided) y q (excluded) are even.
    2. (0.5 points) (Specification, NO implementation) Using the predicate *allEven*, specify a function that given a vector of positive integers of length ≥ 0, it returns the number of non-empty segments that satisfy that all their elements are pairs.
    3. (2 points) Design and implement an iterative algorithm that solves the proposed problem efficiently.
    4. (0.5 points) Write the loop invariant that allows to prove the correctness of the loop and indicate the termination function.
    5. (0.25 points) Indicate the asymptotic cost of the algorithm in the worst case and adequately justify your answer.

Input

The input starts with a line containing the number of test cases. Each test case will contain the value of the number of n elements of the vector, followed by the elements of the vector.

Output

For each test case the program will write a line with the number of segments requested in the instructions of the exercise.

Input Example

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

Output Example

```
0
3
0
2
10
```

2.(2.5 points) A digit of a natural number n is said to be *multiplicative* if it is equal to the remainder of dividing by 10 the product of the digits that are more significant than it. The most significant digit is multiplicative if it is 1. For example, in the number 23638 there are 2 multiplicative digits, i.e. digit $6 = (2*3)\%10$ and digit $8 = (2*3*6*3)\%10$. You want to count how many multiplicative digits a number has.

Se pide:

1. (1.75 points) Write an efficient recursive algorithm to solve the problem for a given number n. It is not allowed to store in an auxiliary vector the digits of the number.
2. (0.75 points) Write the recurrence corresponding to the cost of the recursive function using the number of digits of n as the size of the problem. Also indicate to which order of asymptotic complexity this cost belongs.

Input

The input begins with a line containing the number of test cases. Each test case will contain the numbera comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba contendrá el número n.

Output

For each test case the program will write the number of multiplicative digits of that case.

Input Example

```
5
0
1
22
65070
23638
```

Output Example

```
0
1
1
2
2
```

3. (4 points) A laboratory mouse has n keys available in its cage to press. When it presses a key for the first time it sets in motion a reward and punishment mechanism. Each time it presses a key again it receives a reward or a punishment depending on the last key it pressed and the current one. This information is given in a matrix T in which for each pair of keys i, j $(0 \leq i, j < n)$ $T[i][j]$ indicates the reward (a value $geq0$) or punishment (a value < 0) that the mouse receives when pressing the j key immediately after the i.

We are asked to design and implement a backtracking algorithm that solves the problem of finding a sequence of keys of length m that maximizes the sum of rewards obtained by the mouse, taking into account that the sum of the penalties in absolute value cannot be greater than a given value $C \geq 0$.

- (3 points) Implement a backtracking algorithm that solves the problem. Explain clearly the markers you have used.
- (1 points) Propose at least one optimality pruning function and implement it in your algorithm.

Input

The input starts with a line containing the number of test cases. Each test case will initially contain the value of the number of keys n, the length of the desired sequence m and the punishment C that must not be exceeded. Then n rows indicate the rewards/punishments that the mouse receives as explained above concerning the matrix T.

Output

For each test case the program will write the highest reward the mouse can get, or NO in case it is not possible to get a sequence of m punishable keystrokes bounded by C.

Input Example

```
3
2 3 -1
-1 -1
-1 -1
2 3 -2
2 3
-1 -2
3 2 0
2 -1 -5
3 2 1
4 -2 2
```

Output Example

```
NO
5
6
```