Problem A: Hoops

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Three children, Alan, Berta and Charlie, go to their school's basketball court to shoot exactly N hoops. Alan wonders how many possibilities there are for who scores which of the N goals, considering that Charlie never passes the ball to Alan and assuming first, that after each attempt to score a goal, whoever made the attempt retrieves the ball themselves, and second, that after obtaining the ball, one scores at least one goal before passing it on. Can you help him?

Input

The input consists of a single line containing an integer N ($0 \leq N \leq 45$) – the number of hoops they want to shoot.

Output

Output the amount of possibilities described above.

Sample input and output

Input	Output
0	1
3	21
11	46368

Problem B: Amusement Parking

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

When you were high on the internet the other night, you accidentally bought a huge amusement park. To be able to settle the enormously high credit card debt you accumulated in the process, you decided to optimize the park's business model.

First of all you noticed that many potential customers can't visit the park because the parking area is way too small to contain all their cars. You already filed an application to expand it, but for now you have to put up with the current size, so you decided to enact some restrictions as to who is allowed to park there from now on. To determine how you should design those rules in order to maximize profit, you analyzed some statistical data about the park's visitors and concluded that families with children are expected to spend 100$ per family member, whereas a couple without children is only expected to spend 150$ in total. (It is guaranteed that no one wants to visit the park unaccompanied.)

The current parking area consists of two continuous stripes without marked parking spots, in which the cars can park directly next to another (door to door). Both are precisely 300m long. One of them can only fit cars of at most 4m length, the other one even those of up to 6m length. Cars longer than that can't be contained and consequently have to be sent away. Because longer cars tend to include more visitors, you decide not to let any cars that also fit on the narrower stripe park on the wider one (therefore any given car is allowed to park on at most one of the stripes).

You are given the lengths, widths and passenger numbers of all cars waiting to enter the parking area today. You need to find out which of them you should let in, in order to maximize the expected turnover of your amusement park according to the statistical data described above (i.e. you need use the car's lengths to determine which stripe they are allowed to park on and their widths and passenger numbers to decide which to let in, because the total width of all cars on a stripe may not exceed 300m). You only need to output the maximum expected turnover.

Input

The input consists of

- one line containing N ($1 \leq N \leq 1000$) – the number of waiting cars
- N lines describing the cars, with the i-th line containing three integers l_i, w_i and a_i ($25 \leq l_i \leq 75, 18 \leq w_i \leq 26, 2 \leq a_i \leq 12$) – the length of the i-th car, the width it occupies when parking (both in decimeters) and the number of passengers in it.

You may assume any car with more than two passengers to contain a family, so that you can apply the known statistical data.

Output

Output maximum expected turnover as described above.

Sample input and output

Input	**Output**
2 51 21 5 39 18 2	650$

Problem C: Longest Alternating Subsequence

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

As you became bored with all the increasing and decreasing sequences, you decided to mix them up and take a look at alternating sequences instead. That is, sequences $s_1, ..., s_k$ that satisfy either $s_1 < s_2 > s_3 < s_4, ...$ or $s_1 > s_2 < s_3 > s_4, ...$ (for example $1, 0, 1, 0, 1, 0, 1$ is an alternating sequence, whereas $1, 1, 0, 1, 0, 1, 0$ isn't). Given an arbitrary sequence, you are to find its longest alternating subsequence.

Input

The input consists of

- one line containing N ($1 \leq N \leq 1000$) - the size of the sequence
- one line containing N numbers $a_1, ..., a_N$ ($-10^4 \leq a_i \leq 10^4$) – the elements of the sequence.

Output

Output the length of the longest alternating subsequence of $a_1, ..., a_N$.

Sample input and output

Input	Output
6 3 29 5 5 28 6	5

Problem D: Casket Of Star

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The Casket of Star [sic] is a device in the Touhou universe. Its purpose is to generate energy rapidly. Initially it contains N stars in a row. The stars are labeled 0 through $N-1$ from the left to the right. Each star has a weight: star i weighs w_i.

The following operation of the device can be repeatedly used to generate energy:

- Choose a star i except 0 and $N-1$.
- The chosen star explodes.
- This generates $w_{i-1} * w_{i+1}$ units of energy.
- Decrease N by one and relabel the stars 0 through $N-1$ from the left to the right.

Your task is to use the device to generate as many units of energy as possible. Return the largest possible total amount of generated energy.

(Hint: $dp[i][j]$ = maximum energy generated using only stars i to j)

Input

The input consists of

- one line containing N ($3 \leq N \leq 50$) – the number of stars
- one line containing N integers $w_1, ..., w_N$ ($1 \leq w_i \leq 1000$), where w_i is the weight of the i-th star.

Output

Output the largest possible total amount of generated energy.

Sample input and output

Input	Output
4 1 2 3 4	12
5 100 2 1 3 100	10400
8 477 744 474 777 447 747 777 474	2937051
7 2 2 7 6 90 5 9	1818
15 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	13

Problem E: Special event

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You have been asked to help organize a special event. You are responsible for the seating of the very important guests. They will all sit in one row, but the problem with that is, that those VIPs are very critical of less important people. As a result they don't want to sit next to people whose importance is less than or equal to half of their own importance. So you need to invite as many important people for the special event as possible and organize the seating such that all of them are happy with it.

Input

The input consists of

- one line containing N ($1 \leq N \leq 1000$) – the number of important people that could be invited
- one line containing integers $a_1, ..., a_N$ ($1 \leq a_i \leq 10^9$) – the importances of these people (in no particular order).

Output

Output the maximal number of important persons that could be invited to the special event and seated in compliance with their preferences.

Sample input and output

Input	Output
5 4 5 2 3 10	4

Problem F: Patents

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The managers of a well-known company have prepared N new patents, because they hope these patents will make their company more powerful. While they did numerate the patents, they didn't do it in any specific order, the patent numbers aren't even unique. But since the company is increasingly said to be a patent troll, the director decides to change the situation for the better. He knows the best way to do this is to make the sequence of patent numbers *convex* (A sequence of numbers $a_1, ..., a_N$ is convex if and only if $a_i \leq (a_{i-1} + a_{i+1})/2$ for all $1 < i < n$).

You are given the sequence $b_1, b_2, ..., b_N$ of patent numbers in the (unchangeable) order prepared by the managers. Your task is to repeal some patents such that the remaining patent numbers $a_1 = b_{k_1}, a_2 = b_{k_2}, ..., a_l = b_{k_l}$ (with some $k_1 < k_2 < ... < k_l$) form a convex sequence. But still the company should keep its strong position, so there should be as many patents as possible left when you are done.

Input

The first line contains T ($1 \leq T \leq 10$) – the number of testcases. Each testcase consists of

- one line containing N ($1 \leq N \leq 200$) – the number of new patents
- one line containing N integers $b_1, ..., b_N$ ($-10^4 \leq b_i \leq 10^4$) – the patent numbers in the order decided on by the managers.

Output

For each testcase output one integer – the maximal possible amount of remaining patents.

Sample input and output

Input	Output
2 3 2 3 5 4 2 3 6 8	3 3

Problem G: Robot Challenge

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You have entered your robot in a Robot Challenge. The course the challenge is taking place on is set up in a 100m by 100m space. Certain points within this space are marked as targets. They are ordered – there is a target 1, a target 2, etc. Your robot must start at the coordinates (0,0). From there, it should go to target 1, stop for 1 second, go to target 2, stop for 1 second, and so on. It must finally end up at, and stop for a second on (100,100).

Each target except (0,0) and (100,100) has a time penalty for missing it. So, if your robot went straight from target 1 to target 3, skipping target 2, it would incur target 2's penalty. Note that once it hits target 3, it cannot go back to target 2, because it must hit the targets in order. Since it has to stop for 1 second on a target point in order to make the interaction count as a hit, the robot is not in danger of accidentally hitting a target too soon. For example, if target point 3 lies directly between target points 1 and 2, your robot can go straight from 1 to 2, right over 3, without stopping. Since it didn't stop, the judges will not mistakenly think that it hit target 3 too soon, so they won't assess target 2's penalty. Your final score is the amount of time (in seconds) your robot takes to reach (100,100), completing the course, plus all penalties. The smaller your score, the better.

Your robot is very maneuverable, but a bit slow. It moves at 1 m/s, but can turn very quickly. During the 1 second it stops on a target point, it can easily turn to face the next target point. Thus, it can always move in a straight line between target points.

Because your robot is a bit slow, it might be advantageous to skip some targets, and incur their penalty, rather than actually maneuvering to them. Given a description of a course, determine your robot's best (lowest) possible score.

Input

There are several test cases. Each test case consists of

- one line containing N ($1 \leq N \leq 1000$) – the number of targets on the course
- N lines describing the targets, with the i-th line containing three integers x_i, y_i and p_i, denoting that the i-th target has coordinates (x_i, y_i) ($1 \leq x_i, y_i \leq 99$, x_i and y_i in meters) and the penalty incurred if the robot misses that target is p_i ($1 \leq p_i \leq 100$). All the targets on a given course will be unique – there will be at most one target point at any location on the course.

The end of the input will be marked by a line with a single 0. It is guaranteed that the sum of N over all testcases doesn't exceed $2 \cdot 10^4$.

Output

For each test case output a single decimal number, indicating the smallest possible score for that course, with precision 10^{-6}.

Sample input and output

Input	**Output**
1 50 50 20 3 30 30 90 60 60 80 10 90 100 3 30 30 90 60 60 80 10 90 10 0	143.4213562 237.7161841 154.4213562