Greetings From Globussoft

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net, PHP

- To solve these 3 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any Search Engine like Google, Yahoo, Bing ...

All the best for your test

Globussoft

QUESTION - 1

Charlie acquired airline transport company and to stay in business he needs to lower the expenses by any means possible. There are N pilots working for his company (N is even) and N/2 plane crews needs to be made. A plane crew consists of two pilots - a captain and his assistant. A captain must be older than his assistant. Each pilot has a contract granting him two possible salaries - one as a captain and the other as an assistant. A captain's salary is larger than assistant's for the same pilot. However, it is possible that an assistant has larger salary than his captain. Write a program that will compute the minimal amount of money Charlie needs to give for the pilots' salaries if he decides to spend some time to make the optimal (i.e. the cheapest) arrangement of pilots in crews.

Input

The first line of input contains integer N, $2 \leq N \leq 10{,}000$, N is even, the number of pilots working for the Charlie's company. The next N lines of input contain pilots' salaries. The lines are sorted by pilot's age, the salaries of the youngest pilot are given the first. Each of those N lines contains two integers separated by a space character, X i Y, $1 \leq Y < X \leq 100{,}000$, a salary as a captain (X) and a salary as an assistant (Y).

Output

The first and only line of output should contain the minimal amount of money Charlie needs to give for the pilots' salaries.

Sample

```
input
4
5000 3000
6000 2000
8000 1000
9000 6000

output
19000

input
6
10000 7000
9000 3000
6000 4000
5000 1000
9000 3000
8000 6000

output
32000
```

QUESTION – 2

A train line has two stations on it, A and B. Trains can take trips from A to B or from B to A multiple times during a day. When a train arrives at B from A (or arrives at A from B), it needs a certain amount of time before it is ready to take the return journey - this is the turnaround time. For example, if a train arrives at 12:00 and the turnaround time is 0 minutes, it can leave immediately, at 12:00.

A train timetable specifies departure and arrival time of all trips between A and B. The train company needs to know how many trains have to start the day at A and B in order to make the timetable work: whenever a train is supposed to leave A or B, there must actually be one there ready to go. There are passing sections on the track, so trains don't necessarily arrive in the same order that they leave. Trains may not travel on trips that do not appear on the schedule.

Input

The first line of input gives the number of cases, **N(1<=N<=100)**. **N** test cases follow.

Each case contains a number of lines. The first line is the turnaround time, **T(0<=T<=60)**, in minutes. The next line has two numbers on it, **NA** and **NB**. **NA** is the number of trips from A to B, and **NB** is the number of trips from B to A**(0<=NA,NB<=100)**. Then there are **NA** lines giving the details of the trips from A to B.

Each line contains two fields, giving the HH:MM departure and arrival time for that trip. The departure time for each trip will be earlier than the arrival time. All arrivals and departures occur on the same day. The trips may appear in any order - they are not necessarily sorted by time. The hour and minute values are both two digits, zero-padded, and are on a 24-hour clock (00:00 through 23:59).

After these **NA** lines, there are **NB** lines giving the departure and arrival times for the trips from B to A.

Output

For each test case, output one line containing "Case #x: " followed by the number of trains that must start at A and the number of trains that must start at B.

Example

```
Input:
2
5
3 2
09:00 12:00
10:00 13:00
11:00 12:30
12:02 15:00
```

```
09:00 10:30
2
2 0
09:00 09:01
12:00 12:02
```

Output:

```
Case #1: 2 2
Case #2: 2 0
```

QUESTION – 3

Byteman has a collection of N squares with side 1. How many different rectangles can he form using these squares?

Two rectangles are considered different if none of them can be rotated and moved to obtain the second one. During rectangle construction, Byteman can neither deform the squares nor put any squares upon any other ones.

Input

The first and only line of the standard input contains one integer N ($1 \leq N \leq 10000$).

Output

The first and only line of the standard output should contain a single integer equal to the number of different rectangles that Byteman can form using his squares.

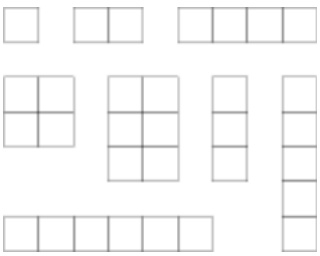
Example

For the input data:

```
6
```

the correct result is:

```
8
```



QUESTION – 4

According to Wikipedia, "The Traveling Salesman Problem (TSP) is a problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once. The problem was first formulated as a mathematical problem in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, a large number of heuristics and exact methods are known, so that some instances with tens of thousands of cities can be solved."

Fortunately, you won't have to solve TSP. You're working for a very clever traveling salesman who has already figured out the path he is going to take. All he needs from you is a quick way to figure out how far he traveled after every segment of his tour.

Input

The salesman kept detailed records of his travels. You'll be getting a series of lines of the form "Some text (X, Y)." indicating that the salesman has been at the point X kilometers east and Y kilometers north of the origin of a Cartesian plane.

Output

For each segment of the trip, output the total distance traveled up to that point as a line in the format "The salesman has traveled a total of D kilometers." Show three digits after the decimal point when printing D. Note that the salesman only travels in straight lines (even after a couple of drinks).

Example

Input:

```
I started out at (0, 5).
Then I traveled to (3.7, 5).
After a couple of drinks I wobbled to (2.7, 4).
The next morning I woke up near (4, 3).
I finished my journey in (-.2, 8).
```

Output:

```
The salesman has traveled a total of 3.700 kilometers.
The salesman has traveled a total of 5.114 kilometers.
The salesman has traveled a total of 6.754 kilometers.
The salesman has traveled a total of 13.284 kilometers.
```

QUESTION – 5

Playing with marbles is one of the king's favorite pastimes. He especially enjoys a game which was taught to him by Eratosthenes, a visiting mathematician from Greece. The rules are very complicated but it all boils down to arranging marbles in a (filled) rectangular shape to score points. If playing with 24 marbles, for example, King Tut could make a 4 by 6 rectangle, or a 3 by 8 rectangle, or a 2 by 12 rectangle. Even the boring 1 by 24 rectangle is allowed. Other numbers of marbles, however, such as 23, make things difficult for the king. Try as he might, the only rectangle he can make is 1 by 23. (Note that rectangles are equivalent under rotation, so a 23 by 1 rectangle would not be a new shape.)

King Tut has decided to call numbers n which can form only the unexciting 1 by n rectangle "non-rectangular." Conversely, numbers like 24, which allow for the formation of more than one rectangle, shall henceforth be referred to as "rectangular." Playing with a single marble is not very interesting at all, so the number 1 will by definition considered neither rectangular nor non-rectangular.

After playing for some time, the king started to notice that every integer greater than 1 could be written as a product of non-rectangular numbers. Were he a mathematician, he would try to prove this claim (which, incidentally, is true). However, the king is more of the engineer type, so he's going to make you verify the claim using brute force! While you're at it, also tell the king how many rectangles can be constructed given a certain number of marbles.

Input

There will be several test cases, one per line, each consisting of a single integer between 2 and 10^7 inclusive. An input of zero will be used to tell your program to stop processing.

Output

For each test case, print out two lines! The first should show the given number of marbles written as a product of non-rectangular numbers, following the example of the sample output. Factors must be written in non-decreasing order and separated by multiplication signs. Also print out spaces around the equals and multiplication signs to improve readability. The second line of output for each test case should be in the format: "With X marbles, Y different rectangles can be constructed." Again, don't forget to replace X and Y with the proper values.

Example

Input:

```
24
23
0
```

Output:

```
24 = 2 * 2 * 2 * 3
With 24 marbles, 4 different rectangles can be constructed.
```

```
23 = 23
With 23 marbles, 1 different rectangles can be constructed.
```
