Given a specified total t and a list of n integers, find all distinct sums using numbers from the list that add up to t. For example, if $t = 4$, $n = 6$, and the list is [4, 3, 2, 2, 1, 1], then there are four different sums that equal 4: 4, 3+1, 2+2, and 2+1+1. (A number can be used within a sum as many times as it appears in the list, and a single number counts as a sum.) Your job is to solve this problem in general.

Input

The input file will contain one or more test cases, one per line. Each test case contains t, the total, followed by n, the number of integers in the list, followed by n integers $x_1, \ldots, x_n$. If $n = 0$ it signals the end of the input; otherwise, t will be a positive integer less than 1000, n will be an integer between 1 and 12 (inclusive), and $x_1, \ldots, x_n$ will be positive integers less than 100. All numbers will be separated by exactly one space. The numbers in each list appear in nonincreasing order, and there may be repetitions.

Output

For each test case, first output a line containing '`Sums of `', the total, and a colon. Then output each sum, one per line; if there are no sums, output the line '`NONE`'. The numbers within each sum must appear in nonincreasing order. A number may be repeated in the sum as many times as it was repeated in the original list. The sums themselves must be sorted in decreasing order based on the numbers appearing in the sum. In other words, the sums must be sorted by their first number; sums with the same first number must be sorted by their second number; sums with the same first two numbers must be sorted by their third number; and so on. Within each test case, all sums must be distinct; the same sum cannot appear twice.

Sample Input

```
4 6 4 3 2 2 1 1
5 3 2 1 1
400 12 50 50 50 50 50 50 25 25 25 25 25 25
0 0
```

Sample Output

```
Sums of 4:
4
3+1
2+2
2+1+1
Sums of 5:
NONE
Sums of 400:
50+50+50+50+50+50+25+25+25+25
50+50+50+50+50+25+25+25+25+25+25
```

You have a long drive by car ahead. You have a tape recorder, but unfortunately your best music is on CDs. You need to have it on tapes so the problem to solve is: you have a tape N minutes long. How to choose tracks from CD to get most out of tape space and have as short unused space as possible.

Assumptions:

- number of tracks on the CD does not exceed 20
- no track is longer than N minutes
- tracks do not repeat
- length of each track is expressed as an integer number
- N is also integer

Program should find the set of tracks which fills the tape best and print it in the same sequence as the tracks are stored on the CD

Input

Any number of lines. Each one contains value N, (after space) number of tracks and durations of the tracks. For example from first line in sample data: $N = 5$, number of tracks=3, first track lasts for 1 minute, second one 3 minutes, next one 4 minutes

Output

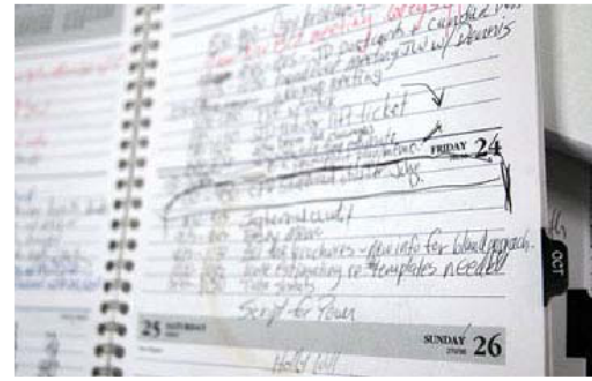
Set of tracks (and durations) which are the correct solutions and string '`sum:`' and sum of duration times.

Sample Input

```
5 3 1 3 4
10 4 9 8 4 2
20 4 10 5 7 4
90 8 10 23 1 2 3 4 5 7
45 8 4 10 44 43 12 9 8 2
```

Sample Output

```
1 4 sum:5
8 2 sum:10
10 5 4 sum:19
10 23 1 2 3 4 5 7 sum:55
4 10 12 9 8 2 sum:45
```



Calendars control our daily lives. For people like me, who are bad at multitasking, it is important to have at most one task planned for any minute of my life. Your job is to make sure that my calendar is free of conflicts for the next one million minutes (just over 99 weeks) of my life. To keep things simple, all times are expressed in minutes from a fixed time 0 representing "now".

In this calendar, there are two types of tasks: one-time tasks and repeating tasks. One-time tasks have a start time and an end time. Repeating tasks have a start time and an end time for their first occurrence, and a repetition interval. Repeating tasks are assumed to keep repeating forever without end. For example, a repeating task with start time 5, end time 8 and repetition interval 100 would be occurring at time intervals [5..8], [105..108], [205..208], ...

Tasks are considered to be in conflict if and only if their time intervals overlap, for example [2..5] and [4..6] overlap. "Touching" is OK, for example [2..5] and [5..6] do not overlap.

Input

There are approximately 30 test cases. The first line of each test case contains two numbers n and m. n is the number of one-time tasks and m the number of repeating tasks. The following n lines contain two numbers each, the start and end times respectively of a one-time task. Afterward, m more lines similarly describe the repeating tasks by giving their start times, end times, and repetition intervals. Both n and m are at most 100.

All numbers are integers in the range [0..1000000]. For each task, the end time is guaranteed to be larger than the start time, and the repetition interval is larger than 0.

Input terminates with a line containing '`0 0`' which should not be processed.

Output

For each test case, print a single line containing either the words '`NO CONFLICT`' if there are no overlaps between any tasks for minutes 0..1000000, or '`CONFLICT`' if there is at least one overlap.

Sample Input

```
2 0
10 20
20 30
2 0
10 30
20 21
1 1
1000 2000
0 10 1000
0 0
```

Sample Output

```
NO CONFLICT
CONFLICT
CONFLICT
```

A tug of war is to be arranged at the local office picnic. For the tug of war, the picnickers must be divided into two teams. Each person must be on one team or the other; the number of people on the two teams must not differ by more than 1; the total weight of the people on each team should be as nearly equal as possible.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of input contains n the number of people at the picnic. n lines follow. The first line gives the weight of person 1; the second the weight of person 2; and so on. Each weight is an integer between 1 and 450. There are at most 100 people at the picnic.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Your output will be a single line containing 2 numbers: the total weight of the people on one team, and the total weight of the people on the other team. If these numbers differ, give the lesser first.

Sample Input

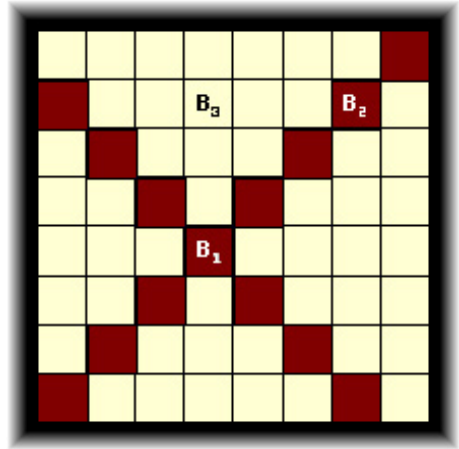
```
1

3
100
90
200
```

Sample Output

```
190 200
```

A bishop is a piece used in the game of chess which can only move diagonally from its current position. Two bishops attack each other if one is on the path of the other. In the figure below, the dark squares represent the reachable locations for bishop B_1 from its current position. Bishops B_1 and B_2 are in attacking position, while B_1 and B_3 are not. Bishops B_2 and B_3 are also in non-attacking position.



Given two numbers n and k, determine the number of ways one can put k bishops on an $n \times n$ chessboard so that no two of them are in attacking positions.

Input

The input file may contain multiple test cases. Each test case occupies a single line in the input file and contains two integers n ($1 \leq n \leq 8$) and k ($0 \leq k \leq n^2$).

A test case containing two zeros terminates the input.

Output

For each test case, print a line containing the total number of ways one can put the given number of bishops on a chessboard of the given size so that no two of them lie in attacking positions. You may safely assume that this number will be less than 10^{15}.

Sample Input

```
8 6
4 4
0 0
```

Sample Output

```
5599888
260
```