Final Exam

Sessions Given

- Monday, May 18, 8:00am - 9:50am EST
- Monday, May 18, 12:00 - 1:50pm EST

This exam is:

- open notes/books/any printed resources
- open laptop (anything that you have on your own computer, not online storage)

Online resources that you are allowed to access:

- Gradescope
- course website
- language documentation:
    - Java: https://docs.oracle.com/javase/10/docs/api/
    - C++: https://www.cplusplus.com/reference/ and/or https://en.cppreference.com/w/
- Piazza, https://piazza.com/class/k5fm7e6zdav5s9 , for asking questions during the exam.

Instructions:

Solve three out of the four problems given on the next pages. You will **not** get extra credit for solving all four problems. If you do attempt four problems, we will pick the three with the highest scores. For each problem, your last submission counts.

Grading:

Every exam problem is graded out of 10 points. The total exam grade is the weighted sum computed as follows (assume *scoreN* is a score for a particular problem with *score1* >= *score2* >= *score3*):

*exam = 5 * score1 + 3 * score2 + 2 * score3*

The total score for a problem is determined by the maximum of zero and the sum of the scores for individual tests based on their results. The maximum score for each test is determined by *max_score = 10/number_of_tests*.

test outcome	test score
passed test	*max_score*
wrong answer	*- 0.5 max_score*
runtime error	*- 0.5 max_score*
timeout error	*- 0.5 max_score*
presentation error	*0.75 max_score*

Bags Of Sand

As the president of the Student Council in NYU, Christina is in charge of planning events for the freshman welcome week. Since all existing games give unfair advantage to one group of students or another (the ones familiar with the game) she is creating a completely new game.

A team of students is presented with N bags of sand of different weights. The weight of each bag is printed on the outside. The team can pick two bags and combine them into a single bag, then, after labeling the new bag with its weight, return the new bag to the collection of the other N-2 bags. The team keeps doing this until all the sand is in one bag.
Each time the team combines two bags, the number of bags of sand decreases by one. But each time the team combines two bags they pay a cost equal to the sum of the two weights.

The objective is to complete the task while incurring the minimum cost. Christina asks you to help her figure out what the lowest cost for each set of sand bags is.

For example we have three bags with weights 1, 2, and 3. We can combine them as follows:

Strategy 1	Strategy 2	Strategy 3
1 + 2 = 3, cost of 3	2 + 3 = 5, cost of 5	1 + 3 = 4, cost of 4
3 + 3 = 6, cost of 6	1 + 5 = 6, cost of 6	2 + 4 = 6, cost of 6
Total cost = 9	Total cost = 11	Total cost = 10

The lowest cost of combining the sand bags is 9.

Input

The first line of input contains a positive number `N` (`2 <= N <= 5000`) that tells you how many numbers there are to add. The second line of input contains the weights of `N` bags `0 <= n_1, n_2, ..., n_N <= 100,000` .

Output

The minimum total cost of combining the bags of sand followed by a newline.

Example 1

```
Input:
3
1 2 3

Output:
9
```

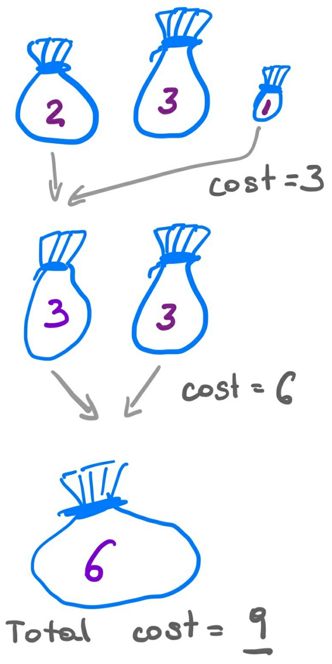

Illustration of Example 1

Example 2

```
Input:
4
1 2 3 4

Output:
19
```

Ayu’s Shopping

Ayu is keen on sweets. She discovered that she will run out of sweets soon so she decided to restock. Her favorite shop offers several flavors (banana, grape, etc.) for each of the N categories of sweets (candy, cake, etc.). She likes all sweets equally so she decided to buy exactly one flavor of each kind of sweet (and flavors can repeat among the categories). Ayu’s parents gave her M dollars and she wants to spend as much of it as possible. Help her figure out the largest amount that she can spend while buying one flavor of sweet in each category.

Input

The first line of the input contains two integers M (1 <= M <= 200) and N (1 <= N <= 20), indicating the amount of money Ayu has and the number of categories of sweets. Each of the following N lines contains K + 1 integers where K is the first integer of that line (1 <= K <= 20), indicating the number of flavors for that kind of sweets. The remaining K integers indicate the price for each flavor.

Output

Print one line consisting of one integer indicating the maximum amount of money needed to buy one flavor in each category of sweets, or `no solution` if there is no solution.

Example 1

```
Input:
100 4
3 8 6 4
2 5 10
4 1 3 3 7
4 50 14 23 8

Output:
75
```

Example 2

```
Input:
20 3
3 4 6 8
2 5 10
4 1 3 5 5

Output:
19
```

Example 3

```
Input:
5 3
3 6 4 8
2 10 6
4 7 3 1 7

Output:
no solution
```

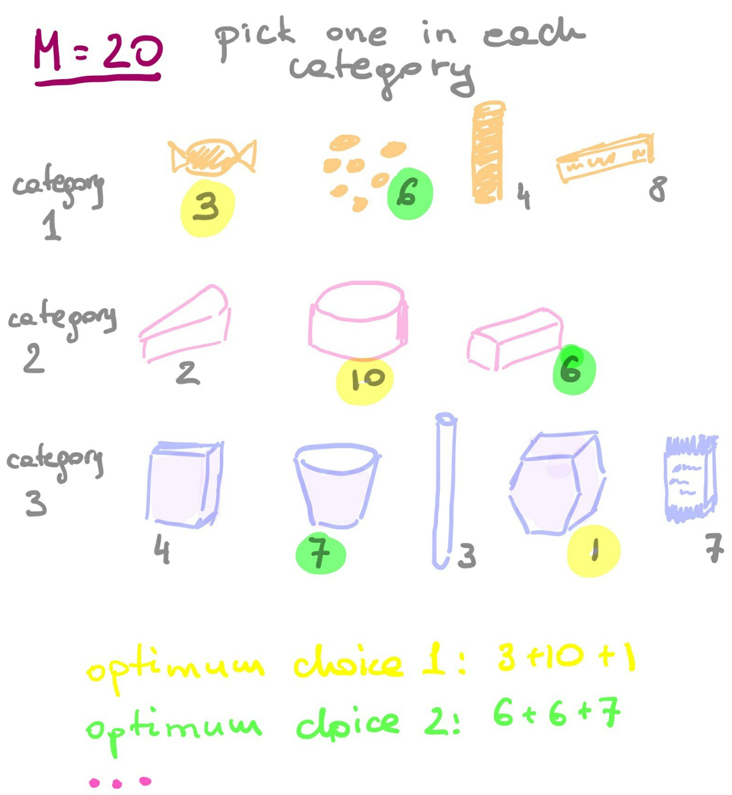

Illustration of Example 2

Task Planning

There are two types of tasks: recurring tasks and one-time tasks. Both types of tasks have a start time and an end time. Additionally, recurring tasks have a repetition interval. Recurring tasks keep repeating forever at a fixed frequency. For example, a repeating task with start time 5, end time 8 and repetition interval 100 will occur at time [5…8], [105…108], [205…208], … You are given N one-time tasks and M repeating tasks. Your job is to determine if there are any conflicts (overlaps) between them in the time interval [0…1,000,000].

Note: tasks are considered to overlap only if their time intervals overlap, but not the endpoint. For example, [2…5] and [4…6] are considered to overlap while [2…4] and [4…6] are not.

Input

The first line of the input contains two integers N and M (0 <= N, M <= 100), indicating the number of one-time tasks and recurring tasks, respectively. Each of the following N lines contains two integers indicating the start time and the end time for the one-time tasks. Afterward, each of the following M lines contains three integers indicating the start time, end time and repetition interval of the recurring tasks.

It is guaranteed that all integers are between 0 and 1,000,000, the end time is larger than the start time and the repetition interval is positive.

Output

Print one line either containing `NO CONFLICT` if there is no conflict / overlap, or `CONFLICT` if there is at least one overlap.

Example 1

```
Input:
2 1
1 3
10 16
2 4 5

Output:
CONFLICT
```

Example 2

```
Input:
2 0
10 15
15 17

Output:
NO CONFLICT
```

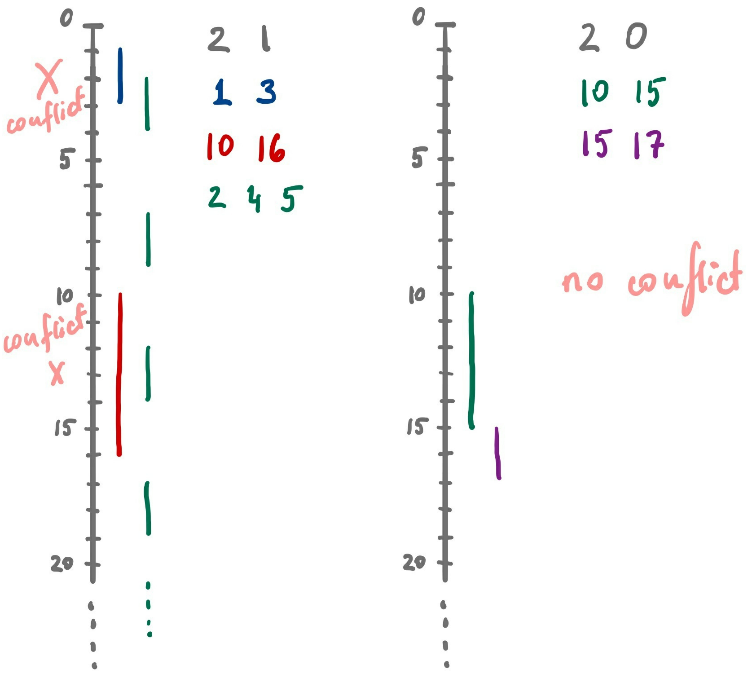

Illustration for example 1 (left) and example 2 (right).

Example 3

```
Input:
1 1
1000 2000
0 10 1000

Output:
CONFLICT
```

Rod Cutting

Christina has another idea for a game for the freshmen welcome week.

Each student team is given a metal rod of length L with markings along the length indicating where the rod needs to be cut (total of N cuts). The cost of cutting the rod at any point is equal to the length of the rod that is being cut. The teams need to decide the order of cuts that minimizes the cost.

For example, if a rod of length 10 feet is to be cut at points 2, 4 and 7 feet from one end, the cuts could be ordered in different ways incurring different costs:

- cut at 2, then at 4 and then at 7 feet, the cost will be 10 + 8 + 6 = 24
- cut at 4, then ant 2 and then at 7, the cost will be 10 + 4 + 6 = 20,
- cut at 7, then at 2 and then at 4, the cost will be 10 + 7 + 5 = 22.

Christina asks you to help her figure out what the lowest cost for rod and a set of cuts is.

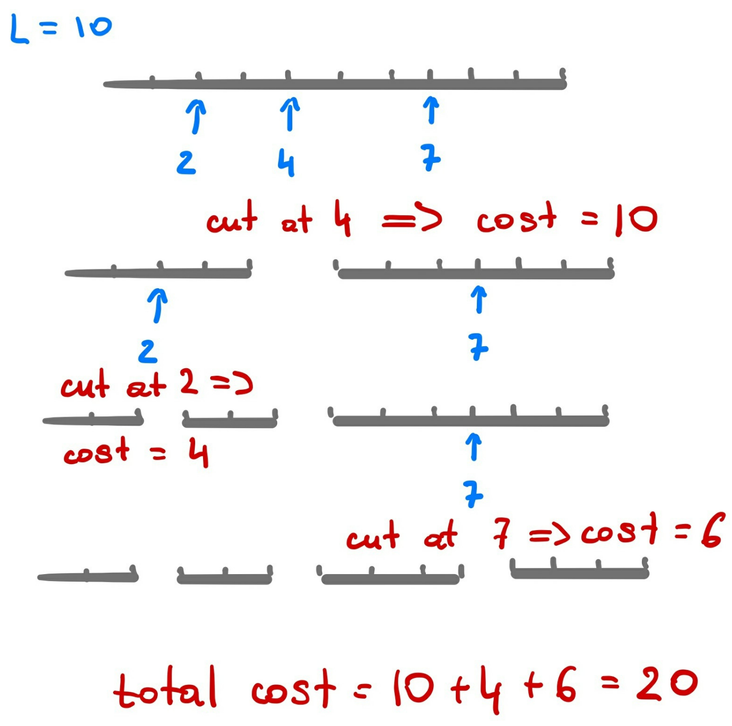

Illustration for example on the left.

Input

The first line of the input contains two integer L (1 <= L < 1,000) and N (1 <= N < 50), representing the length of the rod and the number of cuts to be made, respectively. The second line consists of N integers C_i (1 <= C_i < L), indicating the places where the cuts have to be made. It is guaranteed that C_i are given in increasing order and there is no duplicated C_i.

Output

Print a single line containing a single integer, representing the minimum cost.

Example 1

```
Input:
100
3
25 50 75

Output:
200
```

Example 2

```
Input:
10
4
4 5 7 8

Output:
22
```