Trial A. Candy Sharing Game

Time Limit: 2 seconds

A number of students sit in a circle facing their teacher in the center. Each student initially has an even number of candies. When the teacher blows a whistle, each student simultaneously gives half of his or her candy to the neighbor on the right. Any student, who ends up with an odd number of candies, is given another piece by the teacher. The game ends when all students have the same number of pieces of candy. Write a program which determines the number of times the teacher blows the whistle and the final number of candies for each student from the amount of candy each child starts with.

We note that the game ends in a finite number of steps because:

1. The maximum candy count can never increase.
2. The minimum candy count can never decrease.
3. No one with more than the minimum amount will ever decrease to the minimum.
4. If the maximum and minimum candy count are not the same, at least one student with the minimum amount must have their count increase.

Input:

The input data may describe more than one game. For each game, the input begins with the number N of students, followed by N (even) candies counts for the children counter-clockwise around the circle. The N numbers are on a line and each number is separated by a space. The input ends with a student count of 0. The number N is less than 100 and the initial number of candy is also less than 100.

Output:

For each game, output the number of rounds of the game followed by the amount of candy each child ends up with, both on one line.

Sample input:

```
6
36 2 2 2 2 2
11
22 20 18 16 14 12 10 8 6 4 2
4
2 4 6 8
0
```

Sample output:

```
15 14
17 22
4 8
```

Trial B. Number of Zeros in Factorial

Time Limit: 2 seconds

The factorial of a positive integer n, denoted by $n!$, is the product of all positive integers less than or equal to n. For example, $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$. $0!$ is a special case, it is define to be 1.

For any positive integer n, let $F(n)$ be the number of zeros at the end of the decimal form of the number $n!$. Notice that the function F never decreases. If we have two positive integers $n_1 < n_2$, then $F(n_1) \leq F(n_2)$. It is because we can never lose any trailing zero by multiplying by any positive integer. The function F is very interesting, so we need a computer program that can determine its value efficiently.

Input:

The input data consists of several test cases. Each test case has one line in which contains exactly one positive integer n, here $1 \leq n \leq 2^{30}$. The number 0 indicates the end of the input data.

Output:

For every number n, output a single line containing the non-negative integer $F(n)$.

Sample input:

```
3
60
100
1024
23456
8735373
0
```

Sample output:

```
0
14
24
253
5861
2183837
```

Trial C. Luggage

Time Limit: 2 seconds

Peter and his friends are on holiday. They have decided to make a trip by car to the north of Spain. They are seven people and they think that two cars are enough for their luggage.

It's time to leave and a heap of suitcases are awaiting out of the cars. The drivers disagree about which suitcase must be put into each car, because nobody wants one car to carry more weight than the other one. Is it possible that the two cars load with the same weight? (Obviously without unpacking the suitcases!)

Consider m integers representing suitcases weights, you must decide for each one, if it is possible to distribute the suitcases into the cars, and the two cars carry the same weight.

Input:

Each test case will start with a positive number n, $2 \leq n \leq 20$; followed by n positive integers k_i, $1 \leq k_i \leq 1000$. Input is terminated by a case where the value of n is zero. This case should not be processed.

Output:

For each case print 'YES' or 'NO', depending on the possibility that the two cars will be able to carry the same weight for the respective test case.

Sample input:

```
3 3 4 5
5 1 2 1 2 1
10 2 3 4 1 2 5 10 50 3 50
20 3 5 2 7 1 7 5 2 8 9 1 25 15 8 3 1 38 45 8 1
0
```

Sample output:

```
NO
NO
YES
YES
```

Trial D. Money Change

Time Limit: 2 seconds

A country has k different denominations of coins, $1 = a_1 < a_2 < \ldots < a_k$, where $k \geq 2$ and each a_i is integer, $1 \leq i \leq k$. Charles has a m-dollar bill. He would like to change the dollar bill for coins. That is the sums of money that can be represented using the given coins are then given by

$$m = \sum_{i=1}^{k} a_i x_i,$$

where the x_i are nonnegative integers giving the numbers of each coin used.

Let

$$n = \sum_{i=1}^{k} x_i$$

be the number of coins used. Your task is to write a program to determine the minimum of n, that is the minimum number of coins used.

Input:

The input data contains several test cases. Each test case consists of two lines. The first line contains two integers representing m and k, where $1 \leq m \leq 10000$ and $2 \leq k \leq 100$. The second line contains k integers that denote $a_1, a_2, \ldots, a_k$, where $a_k \leq 2000$. A line with 0 indicates the end of input.

Output:

For each test case, output the minimum of n. The output of each test case should be in a separate line.

Sample input:

```
100 2
1 7
10 3
1 3 4
10000 4
1 5 10 100
0
```

Sample output:

```
16
3
100
```

Trial E. Drainage Ditches

Time Limit: 2 seconds

Every time it rains on Farmer John's fields, a pond forms over Bessie's favorite clover patch. This means that the clover is covered by water for awhile and takes quite a long time to regrow. Thus, Farmer John has built a set of drainage ditches so that Bessie's clover patch is never covered in water. Instead, the water is drained to a nearby stream. Being an ace engineer, Farmer John has also installed regulators at the beginning of each ditch, so he can control at what rate water flows into that ditch. Farmer John knows not only how many gallons of water each ditch can transport per minute but also the exact layout of the ditches, which feed out of the pond and into each other and stream in a potentially complex network.

Given all this information, your task is to write a program to help John to determine the maximum rate at which water can be transported out of the pond and into the stream. For any given ditch, water flows in only one direction, but there might be a way that water can flow in a circle.

Input:

The input includes several cases. For each case, the first line contains two integers m and n ($1 \leq m \leq 200$, $2 \leq n \leq 50$), which are separated by a space. The value of m is the number of ditches that Farmer John has dug. The value of n is the number of intersection points for those ditches. A pair of ditches can only intersect at the intersection point. Intersection point 1 is the pond. Intersection point n is the stream. Each of the following m lines contains three integers, s_i, d_i, and c_i. The values of s_i and d_i designate the intersections between which this ditch flows, where $1 \leq s_i, d_i \leq n$. Water will flow through this ditch from s_i to d_i. The value of c_i is the maximum rate at which water will flow through the ditch, where $0 \leq c_i \leq 10,000$. A line with 0 indicates the end of input.

Output:

For each case, output a single integer that is the maximum rate at which water may emptied from the pond.

Sample input:

```
5 4
1 2 40
1 4 20
2 4 20
2 3 30
3 4 10
5 4
1 2 10
1 3 5
2 3 15
2 4 5
3 4 10
0
```

Sample output:

```
50
15
```