City Evacuation

Things in Area 51 have gone south. Truce between humans and aliens have been called off. Aliens have decided to attack NYC to seek revenge. Emergency has been declared and a secure perimeter has to be laid out. A single UFO's attack can destruct a square area. Unlike regular bombings that destroy a circular area the UFOs cover a square shaped area. Through our satellites it is possible to predict the coordinates from which the UFO's can attack. Considering these coordinates as the center of attack, and given range Ru of attack for an UFO it is possible to find the area that it can destruct: `[Xu - Ru, Xu + Ru]` by `[Yu - Ru, Yu + Ru]`. The satellite department has forwarded you the UFO coordinates and ranges. Find the minimum area in the city to be evacuated such that the are no casualties.

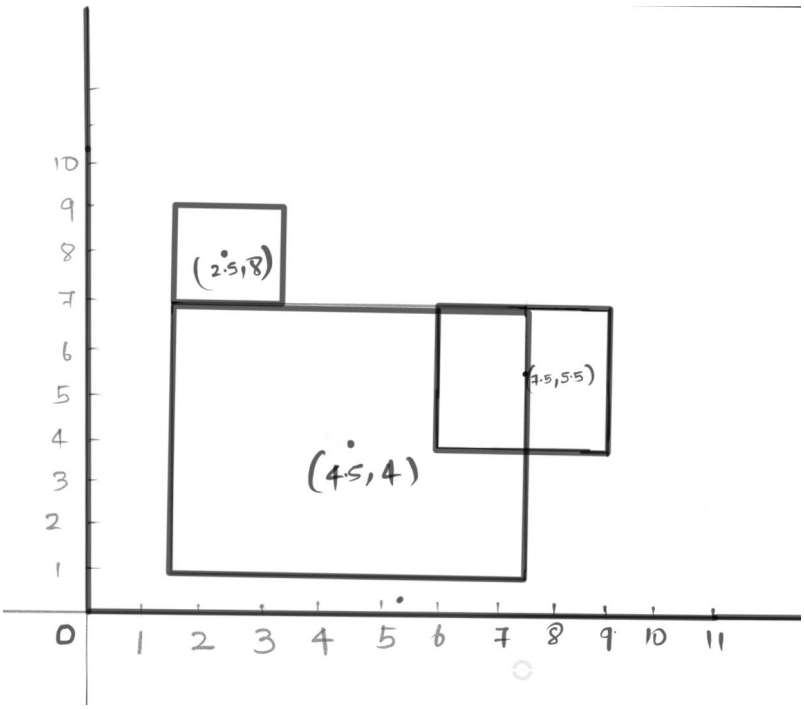

Figure 1:

Input

`N` number of UFOs `1<=N<=100`

`Xu Yu Ru` for each UFOs

`Xu`, `Yu` and `Ru` can be any real value `0<=Xu,Yu,Ru<=200`

Output

The minimum area to be evacuated. Precise to two decimal places.

Example 1

```
Input
3
4.5 4.0 3.0
7.5 5.5 1.5
2.5 8.0 1.0

Output
44.50
```

See image for explanation

Example 2

```
Input
2
2.0 2.0 1.0
3.0 3.0 1.0

Output
7.00
```

Example 3

```
Input
3
5.5 7.5 3.0
6.0 7.0 2.0
4.0 4.0 3.0

Output
60.75
```

Yet Another Sorting Problem (YASP)

Figure 1: "It is lovely day for pancakes."

Kou invented yet another sorting problem (yasp). In this problem, we are given a *stack* of integers and we are asked to sort that stack in ascending order from top to bottom. The only allowed operation, denoted by *flip(t)*, is to flip all elements from the t-th elements to the top upside down. Note that the 1st element is the bottom element while the N-th element refers to the top element if the size of the stack is N. For example, consider the following four stacks of integers.

```
  4       2       5       1       (top)
  3       3       4       2
  2       4       3       3
  5       5       2       4
  1       1       1       5       (bottom)
(a)     (b)     (c)     (d)
```

Performing *flip(3)* on stack (a) will result in stack (b). Similarly, by applying *flip(2)*, stack (b) will become stack (c). Finally, we can apply *flip(1)* on stack (c) and get the stack sorted as shown in stack (d).

We need to figure out a sequence of *flip* operations that transforms the given stack into a sorted one.

Input

The input consists of a single line. The stack of integers will be given in this line from top to bottom. The size of the stack is between 1 and 30, and those integers are between 1 and 100.

Output

Print one line containing (K+1) flip operations:

- Among the first K integers, the i-th integer specifies the t value of the i-th flip operation.
- The last, i.e., the (K+1)-th integer should always be `0`.

If there are multiple solutions, output any of them.

Example 1

```
Input:
1 2 3 4 5

Output:
0
```

Example 2

```
Input:
5 4 3 2 1

Output:
1 0
```

Example 3

```
Input:
5 1 2 3 4

Output:
1 2 0
```

No Change

Chad has been looking for a rare anime garage kit for years.

He just learned about the seller that has exactly what he wants. The seller is a character known for strange rules of trade: she never gives any change.

Figure 1:

The price of the garage kit is M. Chad has brought N coins with him such that the i-th coin has value a_i. He is determined to get the garage kit and is willing to pay more than M, if necessary, but he wants to overpay as little as possible. Moreover, he wants to minimize the number of coins he uses.

Please help him figure out which coins he should use for the purchase.

Input

The first line contains one positive integer M, the price of the garage kit. The price will not exceed 10,000.

The following line contains one positive integer N, the number of coins Chad has, N <= 100.

The next N lines contain positive integers indicating the value of each coin that Chad has. The values are any positive integers no greater than 10,000.

Chad has brought enough money to buy the kit, so the total value of his coins will always be equal to or greater than the price of the kit.

Output

Output a single line containing two integers: the total amount paid and the total number of coins used.

Example 1

```
Input:
14
3
5
10
20

Output:
15 2
```

Example 2

```
Input:
5
3
2
3
5

Output:
5 1
```

Parcels

Anazon has 6 types of products. Each is packed in a box of height H with base equal to one of the following sizes: 1x1, 2x2, 3x3, 4x4, 5x5, 6x6. (All sizes are specified in feet.)

The individual boxed products are always delivered to customers in the shipping packages with height H and base of 6x6 (again in feet).

To reduce the number of shipping packages (and therefore the cost), Anazon wants you to write a program to calculate the minimal number of packages necessary to deliver the given order of products.

Input A single line containing 6 non-negative integer no greater than 100,000, specifies an order.

The i-th integer indicates the number of boxes of individual size i x i.

Output Output the minimum number of packages needed to ship the order.

Example 1

```
Input
0 0 0 0 1 1

Output
2
```

Example 2

```
Input
0 0 4 0 0 1

Output
2
```

Example 2

```
Input
6 5 1 0 0 0

Output
1
```

Max Sum Subarray

In computer science, the maximum sum subarray problem is the task of finding a contiguous subarray with the largest sum, within a given one-dimensional array A[1... n] of numbers.

Formally, the task is to find indices i and j with 1 <= i <= j <= n, such that the sum

a[i] + a[i+1] + ... + a[j-1] + a[j]

is as large as possible.

For example, for the array of values [-2, 1, -3, 4, -1, 2, 1, -5, 4], the contiguous subarray with the largest sum is [4, -1, 2, 1], with sum 6.

Write a program to calculate such sum with given input array.

Input

The first line contains 1 integer n, the number of elements in the input array. 1 <= n <= 100000.

In the next line, there are n integer indicating the elements of the array. We have -10^9 <= $A[i]$ <= 10^9 for 1 <= i <= n.

Output

Output one line containing the value of the sum for the maximum subarray.

Example 1

```
Input:
9
-2 1 -3 4 -1 2 1 -5 4

Output:
6
```

Example 2

```
Input:
4
-3 -2 -4 -1

Output:
-1
```