Lab 3

Problem 1: Longest Increasing Subsequence (LIS)

You are given an array A of n integers. A subsequence of A is a sequence that can be derived from A by deleting some or no elements without changing the order of the remaining elements. An increasing subsequence is one in which the elements are in strictly increasing order. Your task is to find the **length** of the longest increasing subsequence of A.

Input Format

- The first line contains an integer n, the size of the array.
- The second line contains n integers $a_1, a_2, \ldots, a_n$.

Output Format

Output a single integer, the length of the longest increasing subsequence.

Example

Input

```
6
10 22 9 33 21 50
```

Output

```
4
```

Explanation

The longest increasing subsequence is `10 22 33 50`, which has length 4.

Hint

If we know the LIS ending at index $i-1$, we can extend it to compute the LIS ending at index i.

Problem 2: Coin Change – Minimum Coins

You are given m different coin denominations and a target value n. You have an unlimited supply of each denomination. Your task is to determine the minimum number of coins required to make up the value n. If it is not possible to form n using the given coins, output `-1`.

Input Format

- The first line contains two integers m and n.
- The second line contains m integers $c_1, c_2, \ldots, c_m$ ($1 \leq c_i \leq n$), representing the coin denominations.

Output Format

Output a single integer, the minimum number of coins needed to form the sum n. If it is not possible, output `-1`.

Example

Input

```
3 11
1 5 7
```

Output

```
3
```

Explanation

We can form 11 as $5 + 5 + 1$, requiring 3 coins. There is no way to do better.

Hint

The **Coin Change problem** is a classical example of our first paradigm, where the solution for a larger problem can be constructed from the solutions of smaller subproblems. The task is to determine the minimum number of coins required to make a given amount n, using an unlimited supply of coins from a given set of denominations. The key idea is that if we already know the minimum number of coins required to make amounts smaller than n (for example, $n-1$, $n-2$, $\ldots$, $n-c$ where c is a coin denomination), then we can build the solution for n. Specifically, for each amount i from 1 to n, we check all coin denominations c. If $i-c$ is a valid amount, then the solution for i can be obtained as $\text{Numberofcoins}[i] = 1 + \text{Numberofcoins}[i-c]$, where $Numberofcoins[i-c]$ is the optimal solution for the smaller amount. We then take the minimum over all such choices to ensure optimality.

The process starts with the base case $Numberofcoins[0] = 0$ (zero coins are needed to make sum zero). By iteratively building solutions for all values from 1 to n, we finally obtain the minimum number of coins required for n. If no combination of coins can make the sum n, the result is reported as -1. This approach avoids recomputation and ensures that the problem is solved in polynomial time.

Problem 3: Frog Jump — Minimum Energy

A frog is sitting on stone 1 and wants to reach stone n. Each stone i has a height h_i.

- From stone i, the frog can jump either to stone $i + 1$ or to stone $i + 2$ (if they exist).
- The energy cost of jumping from stone i to stone j is given by:

$$|h_i - h_j|$$

Your task is to determine the minimum total energy required for the frog to reach stone n.

Input Format

- The first line contains an integer n, the number of stones.
- The second line contains n integers $h_1, h_2, \ldots, h_n$, where h_i is the height of stone i.

Output Format

Output a single integer, the minimum energy required for the frog to reach stone n.

Example

Input

```
4
10 30 40 20
```

Output

```
30
```

Explanation

The frog jumps from stone $1 \to 2$ (cost $|10 - 30| = 20$), then from stone $2 \to 4$ (cost $|30 - 20| = 10$). The total minimum energy required is 30.