**Q1. Decay Trail Optimization Problem**

**You are working with a climate research lab that tracks the decay of biomass on a 2D terrain M. The terrain is divided into a matrix of cells, where each cell value indicates the biomass concentration at that point.**

**Starting from a specific cell, scientist want to analyze how far the decay can propagate recursively, where each propagation can only move adjacent cell (up, down, left, right) that has strictly lower biomass than the current one.your task is to determine the maximum recursive decay trail length starting from given cell.**

**Input Specification**

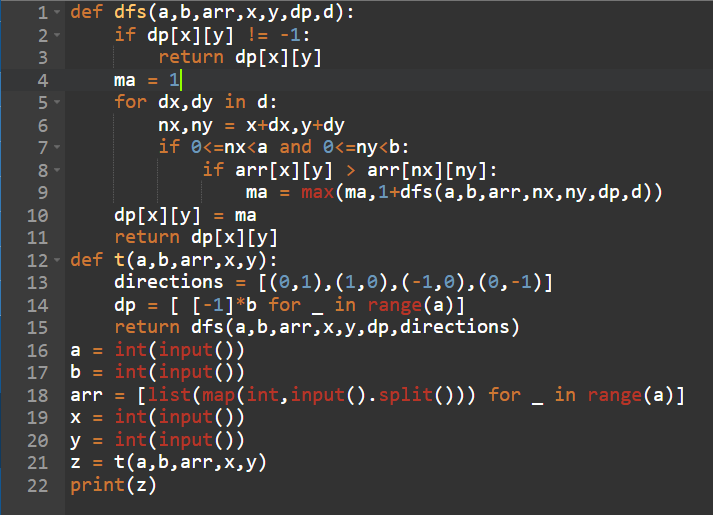
* **input1 (N): An integer representing the number of rows in the matrix.**
* **input2 (M): An integer representing the number of columns in the matrix.**
* **input3 (A 2D array): An N x M integer array representing the terrain's biomass.**
* **input4 (x): An integer representing the starting row index.**
* **input5 (y): An integer representing the starting column index.**

**Output Specification**

* **Return an integer value that represents the maximum recursive decay trail length starting from the given cell (x, y).**

**Example**

* **Input:** 
  + **input1: 3**
  + **input2: 3**
  + **input3: {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}}**
  + **input4: 0**
  + **input5: 0**
* **Output: 5 (The trail starts at (0,0) with value 9 and can potentially follow a path like 9 -> 8 -> 7 -> 5 -> 2 or 9 -> 6 -> 5 -> 4 -> 2 etc. The longest possible path is 5 cells long.)**

****

**Q2. Max Reward Points Problem**

Alex is navigating through a path with **N nodes**, each containing a reward value. He wants to maximize the sum of rewards collected from **non-adjacent nodes** along this path.

However, there's a unique rule: if Alex selects a node and its **immediate predecessor** in the path is negative, he must include **both nodes** in the sum and **subtract the value of the negative node**.

Your task is to help Alex find and return an integer value representing the maximum sum of rewards that can be collected.

**Input Specification**

* **input1:** An integer value denoting the total number of nodes.
* **input2:** An integer array denoting the rewards points of the nodes.

**Output**: Return an integer denoting the max sum of rewards, that Alex can collect as per above conditions.

**Example 1:**

input : 7

input 2: {-1, 2, 5, -3, 10, 6, 1}

Output: 13

**Explanation**: The path that gives the max sum of rewards points is {5, 10, 1} it gives a sum of 13 (-3 is included because it precedes 10). therefore 13 is returned as output.

**Example 2:**

input 1: 7

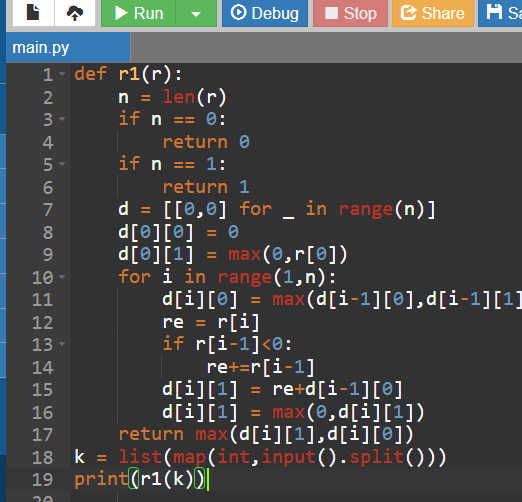
input 2: {-1,2,5,-3,10,9,1}

Output: 14

**Explanation:**

Scenario 1: the path {5,10,1} totals to 16. Since -3 precedes 10, it is included, resulting in adjusted sum of 16-3 =13.

Scenario 2: the path {5,9} totals 14. there are no negative numbers preceding 5 or 9, so the adjusted sum remains 14. For all other scenarios the max sum does not exceed 14. Therefore 14 returned as an output.



**Q3. Prime Transformation Problem**

In a secret cryptographic system, every number must transform into a prime code based on a special digit transformation rule.

You are given a positive integer **N** and you can transform its digits using the following operation:

* In each step, you can choose one digit and **increase or decrease** it by **1** (for e.g., 4 → 3 or 4 → 5).
* You can perform at most **K** such single-digit operations across the entire number.

After at most K changes, you must check whether any transformed version of N becomes a prime number. Your task is to find and return an integer value representing the **smallest possible prime number** obtainable from N using at most K single-digit (±1) changes. If no such transformation leads to a prime number, return **-1**.

**Note:** Leading zeros are not allowed in the transformed version of N.

Input 1: An integer value N, that you must transform in to prime number

input 2: An integer value K, denoting the max number of digit operations you can perform.

Output: Return an integer value representing the smallest possible prime numbers obtainable from N using the most K single digit (+ or - 1) changes. If no transformation, return -1.

**Example 1:**

input 1: 234

input 2: 2

Output: 223

**Explanation:**

Here, N=234, K=2 from 234, we can change 4→3 (which will take 1 operation), so modified N will be 233, it is prime.

In second operation, change 3→2 so modified N will be 223, it is also prime and is smaller than 233.

223 is smallest possible prime number obtained, therefore 223 is returned as output.

**Example 2:**

input 1: 111

input 2: 1

Output: 101

**Explanation**:

N=111, K=1, we can change the middle digit 1→0 or 2, so the variations will be

011 (Invalid, starts with 0)

211 (prime)

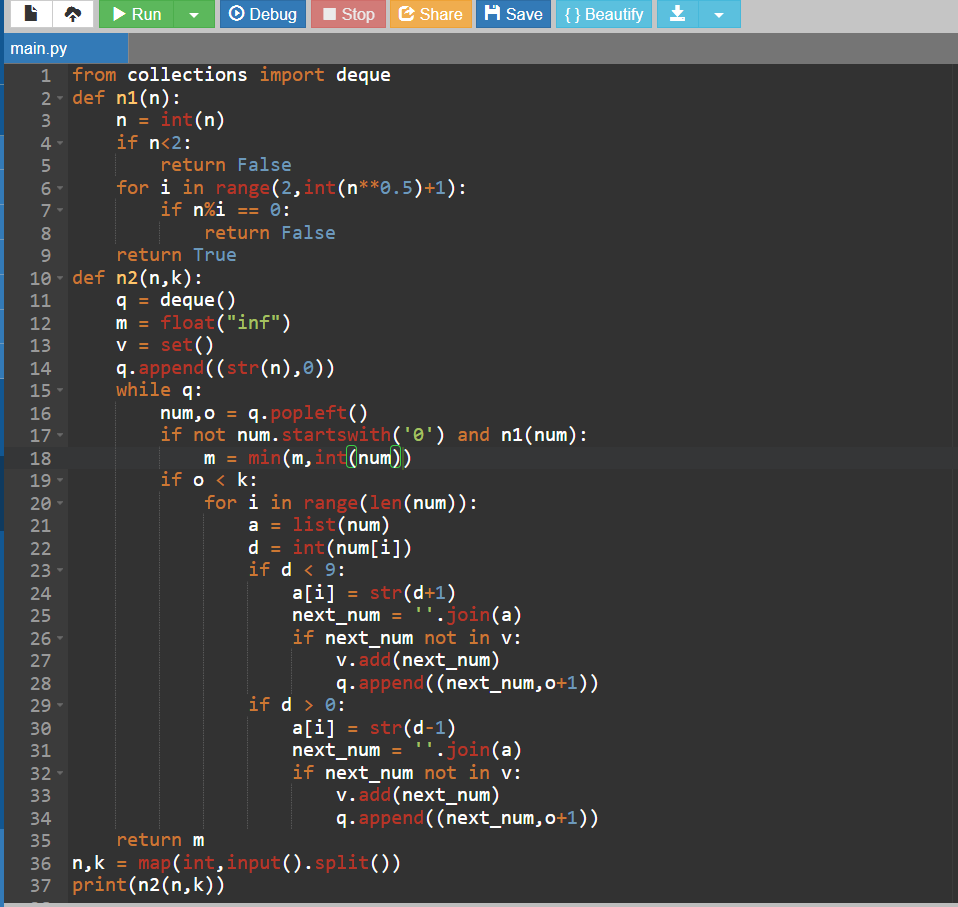
101 (prime)

110 (not prime)

121 (not prime)

112 (not prime)

The smallest prime with at most 1 change is 101, therefore 101 is returned as an output.



**Q5. Minimum Move**

A patient is in a city under quarantine due to a virus outbreak. The city is represented as **N checkpoints in a linear sequence**. Each checkpoint is numbered with a non-negative integer indicating the distance the patient can move from that checkpoint. These distances are stored in an integer array E.

If a patient is at a checkpoint numbered E[i], they can move either E[i] steps to the **right** or E[i] steps to the **left**. However, if moving E[i] steps takes the patient out of the sequence of checkpoints, that move cannot be made.

There is exactly one checkpoint in this sequence that has a hospital providing a life-saving vaccine, which is represented by the number **0**.

Your task is to find and return an integer value representing the **minimum number of moves** the patient requires to reach the hospital, starting from a given checkpoint **K**. If it is not possible for the patient to reach the hospital, return **-1**.

*Note*: Assume **1-based indexing**.

**Input Specification**

* **input1 (N):** An integer value representing the number of checkpoints.
* **input2 (E):** An integer array indicating the distance the patient can move either to the left or to the right from that checkpoint.
* **input3 (K):** An integer value representing the source checkpoint.

Output:

Return an integer value representing the minimum number of moves the patient requires to reach the hospital starting from a given checkpoint K. if not possible for the patient to reach hospital, return -1.

**Example 1:**

input 1: 4

input 2: {2,1,0,1}

input 3: 1

Output : 1

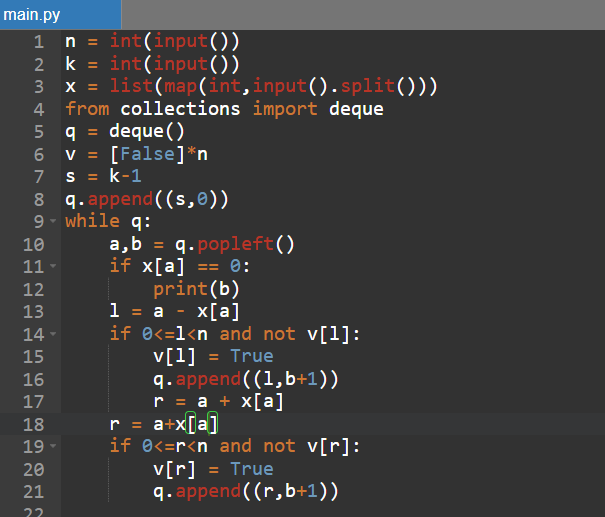
**Example 2:**

input 1: 4

input 2: {3,2,0,2}

input 3: 2

Output : -1



**Q6. LEGO Strings**

Imagine a kid is playing with a special set of LEGO blocks. Each block has a small word written on it, and all blocks are the same size. The kid wants to build a "magic LEGO chain," which is a sequence of some LEGO blocks snapped together in any order, using each block at most once. The goal is to match a part of a wall text with a chain formed by some (more than 1) or all blocks. There should be no gaps, spaces, or extra letters between blocks in the final chain.

You are given:

* A string **S**, called the **wall**, representing a wall of text formed by snapping together many LEGO word blocks.
* An array of **N** words, **B[]**, representing the words written on the available LEGO blocks.

Your task is to find all **starting positions** in the wall where a magic LEGO chain can be formed. This chain must be a **contiguous substring** of the wall, and it must be a combination of blocks from B[] (each used at most once), appearing in some order, joined together without any spaces or gaps.

Return a list of all such starting indices (0-based). If no such position exists, return -1.

**Input Specification**

* **Input 1 (S):** A string representing the wall.
* **Input 2 (N):** An integer value representing the number of blocks.
* **Input 3 (B[]):** A string array representing the LEGO blocks.

Output:

return an integer array representing the starting positions of blocks.

**Example:**

input1: thespiderman

input2: 3

input3: {the, der, esp}

output: {-1}

**Explanation:**

In the scenario we have 3 blocks from which we can make below combinations.

theder, derthe

deresp,espder

theesp,espthe

thedesresp, theespder, dertheesp, esptheder, espderthe

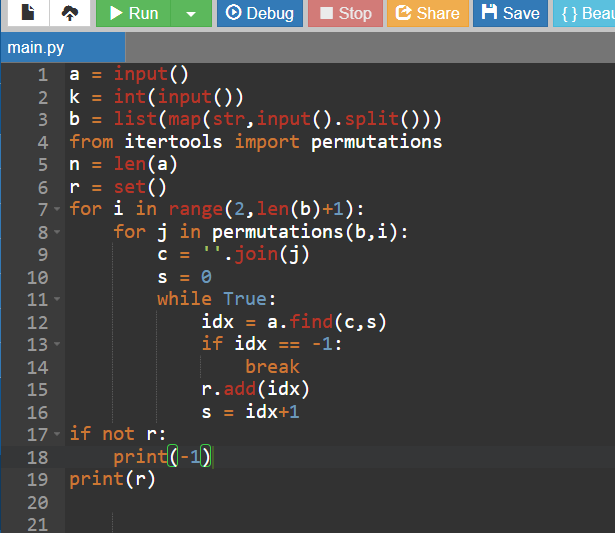
Example 2:

input 1: thespiderman

input 2: 4

input 3: {th, id, es, er}

Output: {0,5}



**Q7. Robot Transmission Filter**

In a high-tech robotics lab, the robot **XR-17** is analyzing a sequence of **N signal nodes** to study network resilience. To do this, XR-17 performs the following operations on the signal sequence:

1. It selects a **K-length consecutive block** of signal nodes to remove from the sequence.
2. It then removes the selected nodes.
3. After removal, it merges the remaining nodes into a new signal pattern.
4. This process is repeated for all possible K-length consecutive blocks in the original sequence.

Given an array **A** representing the N signal node values, your task is to help XR-17 compute how many **distinct signal patterns** can be formed by performing the above process.

*Note:* The signal values are integers ranging from **1 to 20**.

**Input Specification**

* **input1 (N):** An integer value representing the total number of signal nodes.
* **input2 (K):** An integer value representing the length of the consecutive block to be removed.
* **input3 (A):** An integer array representing the signal node values.

Output:

Return an integer value representing the total number of distinct signal patterns XR-17 can generate.

Example 1:

input 1: 4

input 2: 2

input 3: {5,6,1,2}

Output: 3

Example 2:

input 1: 7

input 2: 3

input 3: {17,5,8,12,10,8,5}

with 7 signal nodes and a removal block size of 3, the resulting patterns are

remove indices

0-2 (17,5,8) → remaining {12,10,8,5}

1-3(5,8,12) → remaining {17,10,8,5}

2-4(8,12,10) → remaining {17,5,8,5}

3-5(12,10,8) → remaining {17,5,8,5} (duplicate)

4-6 (10,8,5) → remaining {17,5,8,12}

there are 4 unique patterns among these, 4 is returned as output.



1.additional codes:

**🔹 Question 1: Waste of Time (Recursion)**

**Problem Statement:**

Sid is learning combinations in his class. His teacher has assigned him a task in which a string containing letters from 'a' to 'h' will be given to him. Each letter is mapped to a set of numbers. Mapping of digits to letters is given below:

a => [1,2]

b => [3,4]

c => [5,6]

d => [7,8]

e => [9,10,11]

f => [12,13]

g => [14,15]

h => [16]

He can create a 2D array consisting of 1D arrays having all the possible combinations of the mapped numbers of the string.

Your task is to find and return the resultant sum by adding all the numbers from the possible arrays.

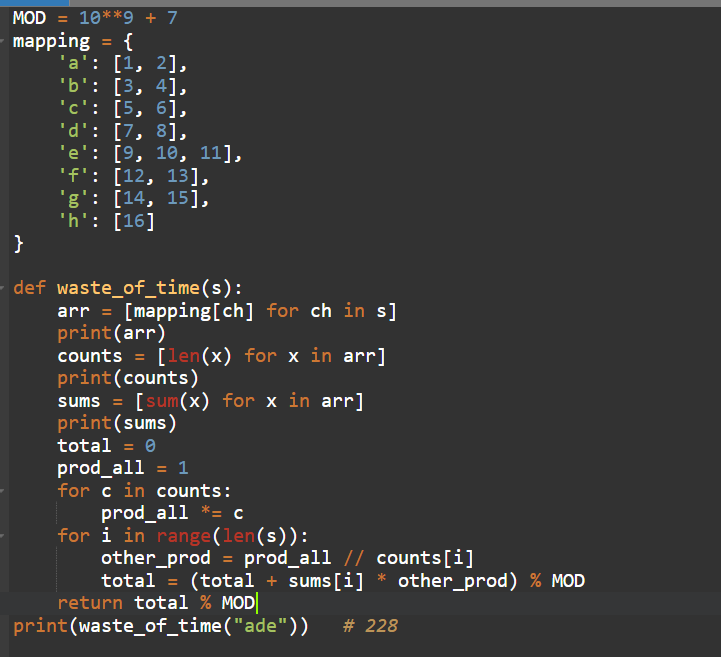
**Note:** Length of the arrays should be equal to length of the input string. As the sum can be very large, return sum modulo (10^9 + 7).

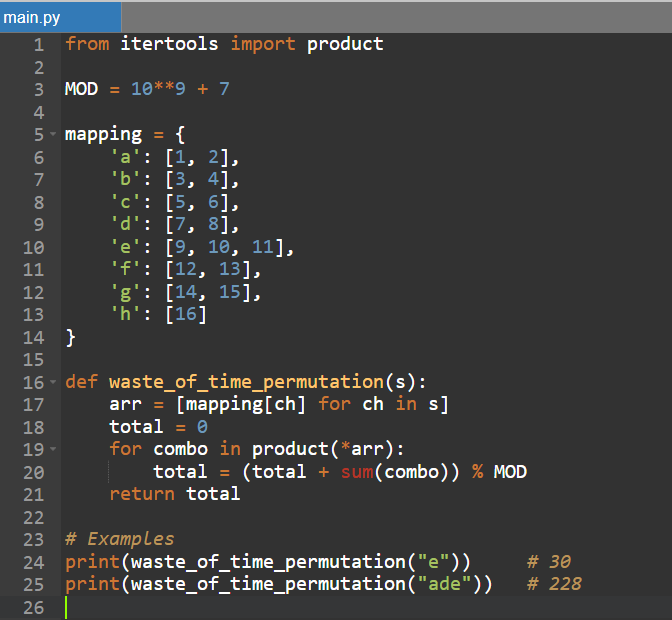
**Input Specification:** input1: A string value representing the string given to Sid by his teacher.

**Output Specification:** Return an integer value representing the sum of all the numbers from the possible arrays.

**Example 1:** input1: e **Output**: 30

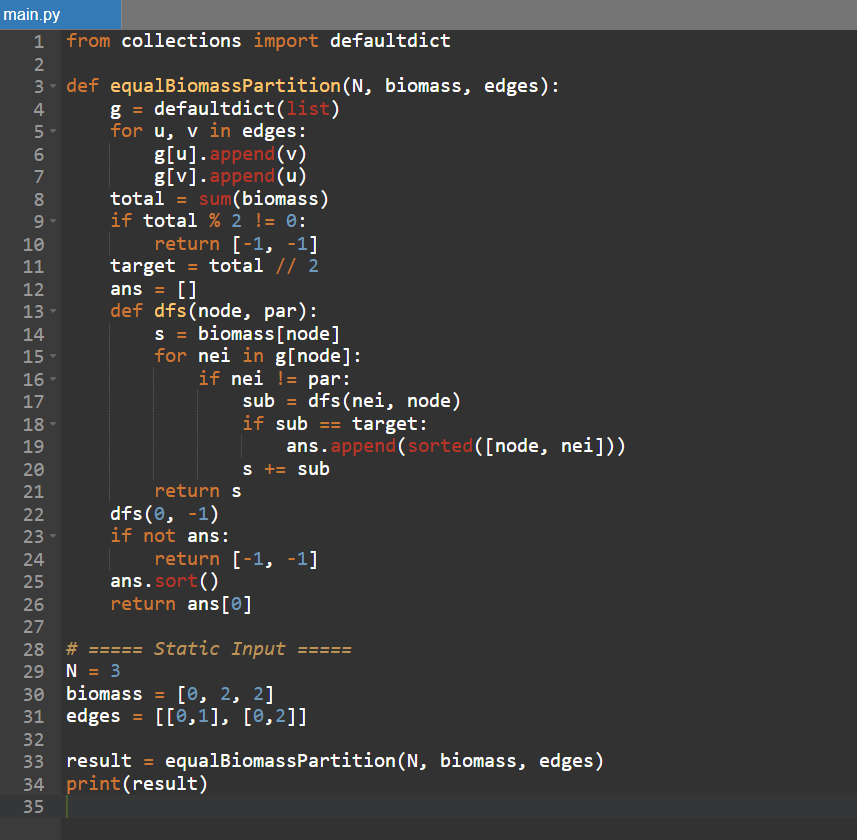
**Example 2:** input1: ade **Output**: 228







Biomass:



Lego 