**Buổi 1**

**Bài 1:** Tính và in ra tích của 10 số tự nhiên đầu tiên . [0 →10].

**Bài 2:** Nhập vào số nguyên dương n từ bàn phím. Tính và in ra n! (giai thừa).

**Bài 3:** Nhập vào số nguyên dương n từ bàn phím. Kiểm tra xem số n có phải là số nguyên tốt hay không. Nếu là số nguyên tố thì in ra dòng text : “Đây là số nguyên tố”. Nếu không thì in ra: “Không phải là số nguyên tố”.

**Bài 4:** Nhập vào từ bàn phím số nguyên n. In ra tổng của các số thỏa mã hai điều kiện:

- nhỏ hơn n và là số chẵn.

**Bài 5:** nhập vào số nguyên từ bàn phím. In ra tích của 2 nhân với các giá trị nhỏ n.

**Input:** 4

**Output:** 2 = 2 \* 1

4 = 2 \* 2

6 = 2 \* 3

8 = 2 \* 4

**Bài 6:** In ra các số trong khoảng từ 10 → 50 thỏa mã điều kiện chia hết cho 2 , vừa chia hết cho 3.

**Bài 7:** Viết một chương trình chấp nhận chuỗi là các dòng được nhập vào, chuyển các dòng này thành chữ in hoa và in ra màn hình. Giả sử đầu vào là:

**Example 1:**

**Input:**

Hello world  
Practice makes perfect

**Output:**

HELLO WORLD  
PRACTICE MAKES PERFECT

Gợi ý: dùng s.upper()

**Bài 8:** Viết một chương trình chấp nhận đầu vào là một chuỗi các từ tách biệt bởi khoảng trắng, loại bỏ các từ trùng lặp rồi in chúng.

**Example 1:**

**Input :** hello world and practice makes perfect and hello world again

**Output:** hello world and practice makes perfect again

**Bài 9:** Đề bài: Yêu cầu người dùng cung cấp một chuỗi và cho biết đó có phải một palindrome không

chú ý: palindrome là một chuỗi có thể được viết xuôi hay viết ngược vẫn chỉ cho ra chính nó).

**Example 1:**

**Input: abccba**

**Output: YES**

**Bài 10:**Viết một chương trình chấp nhận chuỗi từ được phân tách bằng khoảng trống và in các từ chỉ gồm chữ số.

**Example 1:**

**Input:** 3 quantrimang.com và 2 python.

**Output:** [‘3’, ‘2’]

gợi ý dùng: import re

re.findall(“\d+”,s))

**Bài 11: Roman to Integer**

**Symbol** **Value**

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

**Example 5:**

**Input:** s = "MCMXCIV"

**Output:** 1994

→ 1000 + 900 + 90 + 4 = 1994

**Explanation:** M = 1000, CM = 900, XC = 90 and IV = 4.

**Example 1:**

**Input:** s = "III"

**Output:** 3

**Example 2:**

**Input:** s = "IV"

**Output:** 4

**Example 3:**

**Input:** s = "IX"

**Output:** 9

**Example 4:**

**Input:** s = "LVIII"

**Output:** 58

**Explanation:** L = 50, V= 5, III = 3.

**Example 5:**

**Input:** s = "MCMXCIV"

**Output:** 1994

→ 1000 + 900 + 20 100

**Explanation:** M = 1000, CM = 900, XC = 90 and IV = 4.

**Constraints:**

* 1 <= s.length <= 15
* s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').
* It is **guaranteed** that s is a valid roman numeral in the range [1, 3999].

**Bài 12:** [**Decode String**](https://leetcode.com/problems/decode-string)

Given an encoded string, return its decoded string.

The encoding rule is: k[encoded\_string], where the encoded\_string inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.

You may assume that the input string is always valid; No extra white spaces, square brackets are well-formed, etc.

Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there won't be input like 3a or 2[4].

**Example 1:**

**Input:** s = "3[a]2[bc]"

**Output:** "aaabcbc"

**Example 2:**

**Input:** s = "3[a2[c]]"

**Output:** "accaccacc"

**Example 3:**

**Input:** s = "2[abc]3[cd]ef"

**Output:** "abcabccdcdcdef"

**Example 4:**

**Input:** s = "abc3[cd]xyz"

**Output:** "abccdcdcdxyz"

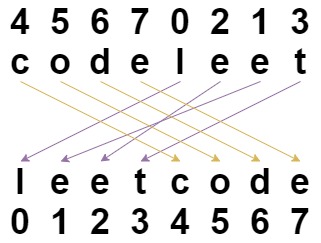
**Bài 13: Shuffle String**

Given a string s and an integer array indices of the **same length**.

The string s will be shuffled such that the character at the ith position moves to indices[i] in the shuffled string.

Return the shuffled string.

**Example 1:**



**Input:** s = "codeleet", indices = [4,5,6,7,0,2,1,3]

**Output:** "leetcode"

**Explanation:** As shown, "codeleet" becomes "leetcode" after shuffling.

**Example 2:**

**Input:** s = "abc", indices = [0,1,2]

**Output:** "abc"

**Explanation:** After shuffling, each character remains in its position.

**Example 3:**

**Input:** s = "aiohn", indices = [3,1,4,2,0]

**Output:** "nihao"

**Example 4:**

**Input:** s = "aaiougrt", indices = [4,0,2,6,7,3,1,5]

**Output:** "arigatou"

**Example 5:**

**Input:** s = "art", indices = [1,0,2]

**Output:** "rat"

**Constraints:**

* s.length == indices.length == n
* 1 <= n <= 100
* s contains only lower-case English letters.
* 0 <= indices[i] < n
* All values of indices are unique (i.e. indices is a permutation of the integers from 0 to n - 1).