💡 \*\*Question 1\*\*

Given an integer n, return *true if it is a power of two. Otherwise, return false*.

An integer n is a power of two, if there exists an integer x such that n == 2x.

**Example 1:** Input: n = 1

Output: true

**Example 2:** Input: n = 16

Output: true

**Example 3:** Input: n = 3

Output: false

**def** is\_power\_of\_two(n):

**if** n **<=** 0:

**return** **False**

**while** n **%** 2 **==** 0:

n **/=** 2

**return** n **==** 1

*# Test Cases*

print(is\_power\_of\_two(1)) *# Output: True*

print(is\_power\_of\_two(16)) *# Output: True*

print(is\_power\_of\_two(3)) *# Output: False*

*#OUTPUT-*

True

True

False

💡 \*\*Question 2\*\*

Given a number n, find the sum of the first natural numbers.

**Example 1:**

Input: n = 3

Output: 6

**Example 2:**

Input : 5

Output : 15

**def** sum\_of\_natural\_numbers(n):

**return** (n **\*** (n **+** 1)) **//** 2

*# Test Cases*

print(sum\_of\_natural\_numbers(3)) *# Output: 6*

print(sum\_of\_natural\_numbers(5)) *# Output: 15*

*#OUTPUT-*

6

15

💡 \*\*Question 3\*\*

\*\*\*\*Given a positive integer, N. Find the factorial of N.

**Example 1:**

Input: N = 5

Output: 120

**Example 2:**

Input: N = 4

Output: 24

**def** factorial(n):

**if** n **==** 0 **or** n **==** 1:

**return** 1

**else**:

**return** n **\*** factorial(n **-** 1)

*# Test Cases*

print(factorial(5)) *# Output: 120*

print(factorial(4)) *# Output: 24*

*#OUTPUT-*

120

24

💡 \*\*Question 4\*\*

Given a number N and a power P, the task is to find the exponent of this number raised to the given power, i.e. N^P.

**Example 1 :**

Input: N = 5, P = 2

Output: 25

**Example 2 :** Input: N = 2, P = 5

Output: 32

**def** power\_of\_number(n, p):

**return** n **\*\*** p

*# Test Cases*

print(power\_of\_number(5, 2)) *# Output: 25*

print(power\_of\_number(2, 5)) *# Output: 32*

*#OUTPUT-*

25

32

💡 \*\*Question 5\*\*

Given an array of integers **arr**, the task is to find maximum element of that array using recursion.

**Example 1:**

Input: arr = {1, 4, 3, -5, -4, 8, 6}; Output: 8

**Example 2:**

Input: arr = {1, 4, 45, 6, 10, -8}; Output: 45

**def** find\_maximum(arr):

**if** len(arr) **==** 1:

**return** arr[0]

**else**:

**return** max(arr[0], find\_maximum(arr[1:]))

*# Test Cases*

arr1 **=** [1, 4, 3, **-**5, **-**4, 8, 6]

arr2 **=** [1, 4, 45, 6, 10, **-**8]

print(find\_maximum(arr1)) *# Output: 8*

print(find\_maximum(arr2)) *# Output: 45*

*#OUTPUT-*

8

45

💡 \*\*Question 6\*\*

Given first term (a), common difference (d) and a integer N of the Arithmetic Progression series, the task is to find Nth term of the series.

**Example 1:**

Input : a = 2 d = 1 N = 5 Output : 6 The 5th term of the series is : 6

**Example 2:**

Input : a = 5 d = 2 N = 10 Output : 23 The 10th term of the series is : 23

**def** nth\_term\_of\_ap(a, d, n):

**return** a **+** (n **-** 1) **\*** d

*# Test Cases*

print(nth\_term\_of\_ap(2, 1, 5)) *# Output: 6*

print(nth\_term\_of\_ap(5, 2, 10)) *# Output: 23*

*#OUTPUT-*

6

23

💡 \*\*Question 7\*\*

Given a string S, the task is to write a program to print all permutations of a given string.

**Example 1:**

\***Input:**\*

*S = “ABC”*

\***Output:**\*

*“ABC”, “ACB”, “BAC”, “BCA”, “CBA”, “CAB”*

**Example 2:**

\***Input:**\*

*S = “XY”*

\***Output:**\*

*“XY”, “YX”*

**def** permutations(string):

**if** len(string) **==** 1:

**return** [string]

perms **=** []

**for** i **in** range(len(string)):

char **=** string[i]

remaining **=** string[:i] **+** string[i**+**1:]

**for** perm **in** permutations(remaining):

perms**.**append(char **+** perm)

**return** perms

*# Test Cases*

print(permutations("ABC")) *# Output: ['ABC', 'ACB', 'BAC', 'BCA', 'CAB', 'CBA']*

print(permutations("XY")) *# Output: ['XY', 'YX']*

*#OUTPUT-*

['ABC', 'ACB', 'BAC', 'BCA', 'CAB', 'CBA']

['XY', 'YX']

💡 **\*\***Question 8**\*\***

Given an array, find a product of all array elements**.**

**\*\***Example 1:**\*\***

Input : arr[] **=** {1, 2, 3, 4, 5}

Output : 120

**\*\***Example 2:**\*\***

Input : arr[] **=** {1, 6, 3}

Output : 18

**def** product\_of\_array(arr):

product **=** 1

**for** num **in** arr:

product **\*=** num

**return** product

*# Test Cases*

arr1 **=** [1, 2, 3, 4, 5]

arr2 **=** [1, 6, 3]

print(product\_of\_array(arr1)) *# Output: 120*

print(product\_of\_array(arr2)) *# Output: 18*

*#OUTPUT-* 120

18