**DSA – ASSIGNMENT 9**

💡 **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

**Solution. :-**

* Check if n is less than or equal to 0. If it is, return False, as negative numbers and zero cannot be powers of two.
* Check if n bitwise AND with (n - 1) is equal to 0. This bitwise operation clears the least significant bit of n, leaving all other bits unchanged. If the result is 0, it means n has only one bit set to 1, and therefore, it is a power of two.
* If the result is 0, return True. Otherwise, return False.

**def isPowerOfTwo(n):**

**if n <= 0:**

**return False**

**return (n & (n - 1)) == 0**

**n = 1**

**result = isPowerOfTwo(n)**

**print(result)**

💡 **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

**Solution. :-**

* Calculate the sum using the formula: Sum = (n \* (n + 1)) / 2.
* Return the calculated sum.

**def sumOfFirstN(n):**

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

**n = 3**

**result = sumOfFirstN(n)**

**print(result)**

💡 **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

**Solution. :-**

* Initialize a variable factorial to 1.
* Iterate from 1 to N (inclusive).
  + Multiply factorial by the current number in the iteration.
* After the iteration, factorial will hold the factorial of N.
* Return factorial.

**def factorial(N):**

**factorial = 1**

**for i in range(1, N + 1):**

**factorial \*= i**

**return factorial**

**N = 5**

**result = factorial(N)**

**print(result)**

💡 **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

**Solution. :-**

* Use the power operator \*\* or the pow() function to calculate N raised to the power P.
* Return the calculated value.

**def calculateExponent(N, P):**

**return pow(N, P)**

**N = 5**

**P = 2**

**result = calculateExponent(N, P)**

**print(result)**

💡 **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

**Solution. :-**

* Define a recursive function findMax(arr, start, end) that takes the array arr, the starting index start, and the ending index end as parameters.
* If start is equal to end, it means there is only one element in the subarray. Return that element as the maximum.
* Calculate the middle index mid as (start + end) // 2.
* Recursively find the maximum element in the left subarray by calling findMax(arr, start, mid) and store the result in a variable leftMax.
* Recursively find the maximum element in the right subarray by calling findMax(arr, mid + 1, end) and store the result in a variable rightMax.
* Compare leftMax and rightMax to determine the maximum of the two.
* Return the maximum of leftMax and rightMax as the maximum element in the current subarray.
* In the main function, call findMax(arr, 0, length - 1) to find the maximum element in the entire array.

**def findMax(arr, start, end):**

**if start == end:**

**return arr[start]**

**mid = (start + end) // 2**

**leftMax = findMax(arr, start, mid)**

**rightMax = findMax(arr, mid + 1, end)**

**return max(leftMax, rightMax)**

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

**result = findMax(arr, 0, len(arr) - 1)**

**print(result)**

💡 **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

**Solution. :-**

* Calculate the Nth term using the formula: Nth term = a + (N - 1) \* d.
* Return the calculated Nth term.

**def findNthTerm(a, d, N):**

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

**a = 2**

**d = 1**

**N = 5**

**result = findNthTerm(a, d, N)**

**print(result)**

💡 **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”

**Solution. :-**

* Define a recursive function permuteString that takes the current permutation string, the remaining characters, and a list to store the permutations as parameters.
* If there are no remaining characters, add the current permutation to the list of permutations.
* Iterate over each character in the remaining characters.
  + Append the current character to the current permutation string.
  + Recursively call permuteString with the updated current permutation, the remaining characters without the current character, and the list of permutations.
  + Remove the appended character from the current permutation string to backtrack.
* In the main function, initialize an empty list to store the permutations.
* Call the permuteString function with an empty current permutation, the input string, and the list of permutations.
* Print the list of permutations.

**def permuteString(currPerm, remainingChars, permutations):**

**if len(remainingChars) == 0:**

**permutations.append(currPerm)**

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

**newPerm = currPerm + remainingChars[i]**

**newChars = remainingChars[:i] + remainingChars[i+1:]**

**permuteString(newPerm, newChars, permutations)**

**def printPermutations(S):**

**permutations = []**

**permuteString("", S, permutations)**

**for perm in permutations:**

**print(perm)**

**S = "ABC"**

**printPermutations(S)**

💡 **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

**Solution. :-**

* Initialize a variable product to 1.
* Iterate through each element in the array.
* Multiply product by the current element.
* After the iteration, product will hold the product of all array elements.
* Return product

**def findProduct(arr):**

**product = 1**

**for num in arr:**

**product \*= num**

**return product**

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

**result = findProduct(arr)**

**print(result)**