**Assignment-9 [Recursion]**

**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

class Solution:

    def isPowerOfTwo(self, n: int) -> bool:

        if n == 1:

            return True

        elif n % 2 != 0 or n ==0:

            return False

        else:

            return self.isPowerOfTwo(n/2)

**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 sumOfNaturals(n):

if n ==1 or n == 0:

return n

else:

return n + self.sumOfNaturals(n-1)

**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

CODE:

class Solution:

def factorial (self, N):

if N== 0 or N == 1:

return 1

else:

return N \* self.factorial(N-1)

**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

CODE:

def power(N,P):

if P == 0:

return 1

else:

return N \* power(N,P-1)

**Question 5**

Given an array of integers **arr**, the task is to find the 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

CODE:

Start looking the array from the end and check the elements using max function by reducing the length of array by 1.

def maxnum(arr, n):

if n ==1:

return arr[0]

else:

return max(arr[n-1], maxnum(arr, n-1))

**Question 6**

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

CODE:1

Using For loop:-

nthTerm = a

for i in range(N):

nthTerm = nthTerm + d

print(nthTerm)

CODE:2 Using Recursion

def nthterm(a):

if a == N:

return a+d

else:

a + self.nthterm(a+d)

**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”

Approach :

1. Create left substring
2. Create right substring
3. Add THE ELEMENT to the end of THE string.
4. Ex: …ABC Left substring of A is “ ” and right substring of A is “BC” Therefore we will add “BCA” now similary left and right substring of B -> “CAB”
5. Basically add left substring + right substring + The character

CODE:-

def permute(s,answer):

if len(s) ==0:

print(answer, end = " ")

return

for i in range(len(s)):

ch = s[i]

left\_string= s[0:i]

right\_string = s[i+1:]

result = left\_string + right\_string

permute(result, answer + ch)

#Drivercode

answer = ""

s = "ABC"

print("All possible strings are : ")

permute(s, answer)

**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

ar[] = {1, 2, 3, 4, 5}

#Output : 120

CODE:

n = len(arr)

def product(arr[]):

if n == 0:

return 1

return arr[n] \* self.product(arr[n-1])

**ASSIGNMENT 10 [RECURSION]**

**Question 1**

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

An integer n is a power of three, if there exists an integer x such that n == 3x.

**Example 1:**

Input: n = 27

Output: true

Explanation: 27 = 33

**Example 2:**

Input: n = 0

Output: false

Explanation: There is no x where 3x = 0.

**Example 3:**

Input: n = -1

Output: false

Explanation: There is no x where 3x = (-1).

class Solution:

    def isPowerOfThree(self, n: int) -> bool:

        if n ==1:

            return True

        elif n % 3 != 0 or n ==0:

            return False

        else:

            return self.isPowerOfThree(n/3)

**Question 2**

You have a list arr of all integers in the range [1, n] sorted in a strictly increasing order. Apply the following algorithm on arr:

* Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.
* Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.
* Keep repeating the steps again, alternating left to right and right to left, until a single number remains.

Given the integer n, return *the last number that remains in* arr.

**Example 1:**

Input: n = 9

Output: 6

Explanation:

arr = [1, 2,3, 4,5, 6,7, 8,9]

arr = [2,4, 6,8]

arr = [2, 6]

arr = [6]

**Example 2:**

Input: n = 1

Output: 1

class Solution:

    def lastRemaining(self, n: int) -> int:

        left = True

        remaining = n

        step = 1 # step is the difference between the numbers

        head = 1 # head is the first element in our array

        while remaining>1:

            if left or remaining % 2 == 1:  #mtlb odd numbers hai

                head = head+step

            step = step \*2

            remaining = remaining// 2

            left = not left

        return head

**Question 3**

\*\*\*\*Given a set represented as a string, write a recursive code to print all subsets of it. The subsets can be printed in any order.

**Example 1:**

Input :  set = “abc”

Output : { “”, “a”, “b”, “c”, “ab”, “ac”, “bc”, “abc”}

**Example 2:**

Input : set = “abcd”

Output : { “”, “a” ,”ab” ,”abc” ,”abcd”, “abd” ,”ac” ,”acd”, “ad” ,”b”, “bc” ,”bcd” ,”bd” ,”c” ,”cd” ,”d” }

Solution :

set = "abc"

output - {"", "a","b","c","ab","ac", "bc","abc" }

index=0

curr = " "

def powerset(set,index, curr):

if index == len(set):

return curr

powerset(set,index+1, curr+set[i])

powerset(set, index+1, curr)

**Question 4**

Given a string calculate the length of the string using recursion.

**Examples:**

Input : str = "abcd"

Output :4

Input : str = "GEEKSFORGEEKS"

Output :13

CODE:

a = 0

n = len(str)

def strlength():

If i==n:

Return zero

Return a + strlength(a+1)

**Question 5**

We are given a string S, we need to find the count of all contiguous substrings starting and ending with same character.

**Examples :**

Input : S = "abcab"

Output : 7

There are 15 substrings of "abcab"

a, ab, abc, abca, abcab, b, bc, bca

bcab, c, ca, cab, a, ab, b

Out of the above substrings, there

are 7 substrings : a, abca, b, bcab,

c, a and b.

Input : S = "aba"

Output : 4

The substrings are a, b, a and aba

CODE:

res = 0

for i in range(len(S)):

for j in range(i, len(S)):

if s[i] == s[j]:

result = result +1

return result

output - 7

**Question 6**

The [tower of Hanoi](https://en.wikipedia.org/wiki/Tower_of_Hanoi) is a famous puzzle where we have three rods and **N** disks. The objective of the puzzle is to move the entire stack to another rod. You are given the number of discs **N**. Initially, these discs are in the rod 1. You need to print all the steps of disc movement so that all the discs reach the 3rd rod. Also, you need to find the total moves.

**Note:** The discs are arranged such that the **top disc is numbered 1** and the **bottom-most disc is numbered N**. Also, all the discs have **different sizes** and a bigger disc **cannot** be put on the top of a smaller disc. Refer to the provided link to get better clarity about the puzzle.

**Example 1:**

Input:

N = 2

Output:

move disk 1 from rod 1 to rod 2

move disk 2 from rod 1 to rod 3

move disk 1 from rod 2 to rod 3

3

Explanation:For N=2 , steps will be

as follows in the example and total

3 steps will be taken.

**Example 2:**

Input:

N = 3

Output:

move disk 1 from rod 1 to rod 3

move disk 2 from rod 1 to rod 2

move disk 1 from rod 3 to rod 2

move disk 3 from rod 1 to rod 3

move disk 1 from rod 2 to rod 1

move disk 2 from rod 2 to rod 3

move disk 1 from rod 1 to rod 3

7

Explanation:For N=3 , steps will be

as follows in the example and total

7 steps will be taken.

#Tower of Hanoi

def towerOfHanoi(numbers, start, aux, end):

if numbers == 1: # disc ek hui too

print(f"move disc 1 from {start} rod to {end} rod")

return

towerOfHanoi(numbers-1,start, end, aux):

print(f"move disc {numbers} from {start} to {aux}")

towerOfHanoi(numbers-1,aux,start, end):

print(f"move disc {numbers} from {aux} to {end}")

numbers = 3

def towerOfHanoi(numbers, "A", "B", "C")

**Question 7**

Given a string **str**, the task is to print all the permutations of **str**. A **permutation** is an arrangement of all or part of a set of objects, with regard to the order of the arrangement. For instance, the words ‘bat’ and ‘tab’ represents two distinct permutation (or arrangements) of a similar three letter word.

**Examples:**

Input: str = “cd”

**Output:** cd dc

**Input:** str = “abb”

**Output:** abb abb bab bba bab bba

CODE:

def permute(s,answer):

if len(s) ==0:

print(answer, end = " ")

return

for i in range(len(s)):

ch = s[i]

left\_string= s[0:i]

right\_string = s[i+1:]

result = left\_string + right\_string

permute(result, answer + ch)

OR

result = 0

def permute():

if len(string)==1:

return str

for i in range(len(string)):

n = string.pop(0)

perm = self.permute(str)

for i in perms:

string.append(n)

result.extend(perms)

string.append(n)

return result

**Question 8**

Given a string, count total number of consonants in it. A consonant is an English alphabet character that is not vowel (a, e, i, o and u). Examples of constants are b, c, d, f, and g.

def countconsonants(str):

count = 0

if len(str)==0:

return count

for i in range(len(str)):

if str[i].lower() in "aeiou":

continue

else:

count = count+1

i +=1

result =(countconsonants("Ram"))

print(result)

**Assignment Questions 11**

**Question 1**

Given a non-negative integer x, return *the square root of* x *rounded down to the nearest integer*. The returned integer should be **non-negative** as well.

You **must not use** any built-in exponent function or operator.

* For example, do not use pow(x, 0.5) in c++ or x \*\* 0.5 in python.

**Example 1:**

Input: x = 4

Output: 2

Explanation: The square root of 4 is 2, so we return 2.

**Example 2:**

Input: x = 8

Output: 2

Explanation: The square root of 8 is 2.82842..., and since we round it down to the nearest integer, 2 is returned.

class Solution:

    def mySqrt(self, x: int) -> int:

        left, right = 1, x

        while left <=right:

            mid = (left + right)//2

            if mid\*mid == x:

                return mid

            elif mid\*mid < x:

                left = mid +1

            else:

                right = mid-1

        return right

**Question 2**

A peak element is an element that is strictly greater than its neighbors.

Given a **0-indexed** integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in O(log n) time.

**Example 1:**

Input: nums = [1,2,3,1]

Output: 2

Explanation: 3 is a peak element and your function should return the index number 2.

**Example 2:**

Input: nums = [1,2,1,3,5,6,4]

Output: 5

Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.

Approach :

Since we know that adjacent numbers can not be equal to each other therefore we will compare with right and left side values of the mid and if the nums[mid]< nums[mid+1] we will search in the right side portion, therefore, left = mid+1 and vice versa.

class Solution:

    def findPeakElement(self, nums: List[int]) -> int:

        left, right = 0, len(nums)-1

        while left<=right:

            mid = (left + right)//2

            if (mid>0) and nums[mid]<nums[mid-1]:

                right = mid-1

            elif mid< len(nums)-1 and nums[mid] < nums[mid+1]:

                left = mid+1

            else:

                return mid

**Question 3**

Given an array nums containing n distinct numbers in the range [0, n], return *the only number in the range that is missing from the array.*

**Example 1:**

Input: nums = [3,0,1]

Output: 2

Explanation: n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.

**Example 2:**

Input: nums = [0,1]

Output: 2

Explanation: n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the range since it does not appear in nums.

**Example 3:**

Input: nums = [9,6,4,2,3,5,7,0,1]

Output: 8

Explanation: n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the range since it does not appear in nums.

class Solution:

    def missingNumber(self, nums: List[int]) -> int:

        return sum(range(0,len(nums)+1))-sum(nums)

        # we can even do this with the help of binary num i.e XOR operator; We will take the list of nums and take the XOR with nums array, SO all the same value nums will get canceled and we will be left with only one element that is the missing element.

**Question 4**

Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive.

There is only **one repeated number** in nums, return *this repeated number*.

You must solve the problem **without** modifying the array nums and uses only constant extra space.

**Example 1:**

Input: nums = [1,3,4,2,2]

Output: 2

**Example 2:**

Input: nums = [3,1,3,4,2]

Output: 3

Solution:

class Solution:

    def findDuplicate(self, nums: List[int]) -> int:

        # linked list cycle problem ; floyd's detection algorithm

        slow, fast = 0,0

        while True:

            slow = nums[slow]

            fast = nums[nums[fast]]

            if slow == fast:

                break

        slow2 = 0

        while True:

            slow = nums[slow]

            slow2 = nums[slow2]

            if slow == slow2:

                return slow

**Question 5**

Given two integer arrays nums1 and nums2, return *an array of their intersection*. Each element in the result must be **unique** and you may return the result in **any order**.

**Example 1:**

Input: nums1 = [1,2,2,1], nums2 = [2,2]

Output: [2]

**Example 2:**

Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]

Output: [9,4]

Explanation: [4,9] is also accepted.

class Solution:

    def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:

        m ={}

        if len(nums1)<len(nums2):

            nums1, nums2 = nums2, nums1

        for i in nums1:

            if i not in m:

                m[i] = 1

            else:

                m[i] +=1

        result = []

        for i in nums2:

            if i in m and m[i]:

                m[i] -=1

                if i not in result:

                    result.append(i)

        return result

**Question 6**

Suppose an array of length n sorted in ascending order is **rotated** between 1 and n times. For example, the array nums = [0,1,2,4,5,6,7] might become:

* [4,5,6,7,0,1,2] if it was rotated 4 times.
* [0,1,2,4,5,6,7] if it was rotated 7 times.

Notice that **rotating** an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].

Given the sorted rotated array nums of **unique** elements, return *the minimum element of this array*.

You must write an algorithm that runs in O(log n) time.

**Example 1:**

Input: nums = [3,4,5,1,2]

Output: 1

Explanation: The original array was [1,2,3,4,5] rotated 3 times.

**Example 2:**

Input: nums = [4,5,6,7,0,1,2]

Output: 0

Explanation: The original array was [0,1,2,4,5,6,7] and it was rotated 4 times.

**Example 3:**

Input: nums = [11,13,15,17]

Output: 11

Explanation: The original array was [11,13,15,17] and it was rotated 4 times.

class Solution:

    def findMin(self, nums: List[int]) -> int:

        res = nums[0]

        left = 0

        right = len(nums)-1

        while left<= right:

            if nums[left]<nums[right]:

                res = min(res, nums[left]) #when it is sorted array

            mid = (left+right) // 2        #when not sorted array

            res = min(res, nums[mid])

            if nums[mid] >= nums[left]:

                left = mid+1

            else:

                right = mid-1

        return res

**Question 7**

Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return [-1, -1].

You must write an algorithm with O(log n) runtime complexity.

**Example 1:**

Input: nums = [5,7,7,8,8,10], target = 8

Output: [3,4]

**Example 2:**

Input: nums = [5,7,7,8,8,10], target = 6

Output: [-1,-1]

**Example 3:**

Input: nums = [], target = 0

Output: [-1,-1]

CODE:

class Solution:

    def searchRange(self, nums: List[int], target: int) -> List[int]:

left = 0

        right = len(nums)-1

        while left<=right:

            mid = (left + right)//2

            if nums[mid] == target:

                if mid > 0 and nums[mid] == nums[mid-1]:

                    return [mid-1, mid]

                elif nums[mid] == nums[mid+1]:

                    return [right-mid, mid]

                return [mid, len(mid)-1]

            elif nums[mid]< target:

                left = mid+1

            elif nums[mid]>target:

                right = mid-1

            else:

                return [-1,-1]

        return [-1,-1]

THE ABOVE CODE IS PASSING ONLY 6 TEST CASES

**Question 8**

Given two integer arrays nums1 and nums2, return *an array of their intersection*. Each element in the result must appear as many times as it shows in both arrays and you may return the result in **any order**.

**Example 1:**

Input: nums1 = [1,2,2,1], nums2 = [2,2]

Output: [2,2]

class Solution:

    def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:

        m ={}

        if len(nums1)<len(nums2):

            nums1, nums2 = nums2, nums1

        for i in nums1:

            if i not in m:

                m[i] = 1

            else:

                m[i] +=1

        result = []

        for i in nums2:

            if i in m and m[i]:

                m[i] -=1

                if i not in result:

                    result.append(i)

        return result

# Assignment Questions 15

<aside> 💡 **Question 1**

Given an array **arr[ ]** of size **N** having elements, the task is to find the next greater element for each element of the array in order of their appearance in the array.Next greater element of an element in the array is the nearest element on the right which is greater than the current element.If there does not exist next greater of current element, then next greater element for current element is -1. For example, next greater of the last element is always -1.

**Example 1:**

Input:

N = 4, arr[] = [1 3 2 4]

Output:

3 4 4 -1

Explanation:

In the array, the next larger element

to 1 is 3 , 3 is 4 , 2 is 4 and for 4 ?

since it doesn't exist, it is -1.

**Example 2:**

Input:

N = 5, arr[] [6 8 0 1 3]

Output:

8 -1 1 3 -1

Explanation:

In the array, the next larger element to

6 is 8, for 8 there is no larger elements

hence it is -1, for 0 it is 1 , for 1 it

is 3 and then for 3 there is no larger

element on right and hence -1.

**Question 2**

Given an array **a** of integers of length **n**, find the nearest smaller number for every element such that the smaller element is on left side.If no small element present on the left print -1.

**Example 1:**

Input: n = 3

a = {1, 6, 2}

Output: -1 1 1

Explaination: There is no number at the

left of 1. Smaller number than 6 and 2 is 1.

**Example 2:**

Input: n = 6

a = {1, 5, 0, 3, 4, 5}

Output: -1 1 -1 0 3 4

Explaination: Upto 3 it is easy to see

the smaller numbers. But for 4 the smaller

numbers are 1, 0 and 3. But among them 3

is closest. Similary for 5 it is 4.

**Question 3**

Implement a Stack using two queues **q1** and **q2**.

**Example 1:**

Input:

push(2)

push(3)

pop()

push(4)

pop()

Output:3 4

Explanation:

push(2) the stack will be {2}

push(3) the stack will be {2 3}

pop() poped element will be 3 the

  stack will be {2}

push(4) the stack will be {2 4}

pop()   poped element will be 4

**Example 2:**

Input:

push(2)

pop()

pop()

push(3)

Output:2 -1

**Question 4**

You are given a stack **St**. You have to reverse the stack using recursion.

**Example 1:**

Input:St = {3,2,1,7,6}

Output:{6,7,1,2,3}

**Example 2:**

Input:St = {4,3,9,6}

Output:{6,9,3,4}

**Question 5**

You are given a string **S**, the task is to reverse the string using stack.

**Example 1:**

Input: S="GeeksforGeeks"

Output: skeeGrofskeeG

**Question 6**

Given string **S** representing a postfix expression, the task is to evaluate the expression and find the final value. Operators will only include the basic arithmetic operators like \***, /, + and -**.

**Example 1:**

Input: S = "231\*+9-"

Output: -4

Explanation:

After solving the given expression,

we have -4 as result.

**Example 2:**

Input: S = "123+\*8-"

Output: -3

Explanation:

After solving the given postfix

expression, we have -3 as result.

**Question 7**

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the MinStack class:

* MinStack() initializes the stack object.
* void push(int val) pushes the element val onto the stack.
* void pop() removes the element on the top of the stack.
* int top() gets the top element of the stack.
* int getMin() retrieves the minimum element in the stack.

You must implement a solution with O(1) time complexity for each function.

**Example 1:**

Input

["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[]]

Output

[null,null,null,null,-3,null,0,-2]

Explanation

MinStack minStack = new MinStack();

minStack.push(-2);

minStack.push(0);

minStack.push(-3);

minStack.getMin(); // return -3

minStack.pop();

minStack.top(); // return 0

minStack.getMin(); // return -2

**Question 8**

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after rain.

**Example 1:**

!<https://assets.leetcode.com/uploads/2018/10/22/rainwatertrap.png>

Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]

Output: 6

Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are trapped.

Approach:

Two pointer approach;

maintain 2 variables rightMax and leftMax:

result = min(L,R) – height[i]

class Solution:

    def trap(self, height: List[int]) -> int:

        if not height: return 0

        l, r = 0, len(height)-1

        leftMax, rightMax = height[l], height[r]

        res = 0

        while l<r:

            if leftMax < rightMax:

                l +=1

                leftMax = max(leftMax, height[l])

                res += leftMax - height[l]

            else:

                r -=1

                rightMax = max(rightMax, height[r])

                res += rightMax - height[r]

        return res