**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)