Assignment-10

10 June 2023 16:12

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:

```
class Solution(object):
  def isPowerOfThree(self, n):
    if n<=0:
      return False
    if n==1:
      return True
    while(n!=1):
      if n%3!=0:
        return False
      n=n//3
    return True
```

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]
```

```
class Solution(object):
  def lastRemaining(self, n):
    if(n == 1):
      return 1
    return 2 * ((n // 2 - self.lastRemaining(n // 2)) + 1)
```

Example 2:

```
Input: n = 1
Output: 1
```

Question 4

Given a string calculate length of the string using recursion.

Examples:

Input : str = "abcd" Output :4 Input : str = "GEEKSFORGEEKS" Output :13

```
def string_length(str) :
  if str ==' ':
    return 0
  else :
    return 1 + string_length(str[1:])
```

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” }

```
def powerSet(string, index, curr):
    if index == len(string):
        print(curr)
        return

    powerSet(string, index + 1,
             curr + string[index])
    powerSet(string, index + 1, curr)
```

Question 5

We are given a string S, we need to find 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

```
def countSubstrs(str, i, j, n):
    # base cases
    if (n == 1):
        return 1
    if (n <= 0):
        return 0

    res = (countSubstrs(str, i + 1, j, n - 1)
        + countSubstrs(str, i, j - 1, n - 1)
        - countSubstrs(str, i + 1, j - 1, n - 2))

    if (str[i] == str[j]):
        res += 1
    return res
```

Question 6

The 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 discs 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 the provided link to get a 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.

```
class Solution:
    def toh(self, n, fro, to, aux):
        # Your code here
        if n == 1:
            print("move disk "+str(n)+" from rod "+str(fro)+" to rod "+str(to))
            return 1
        self.toh(n-1,fro,aux,to)
        print("move disk "+str(n)+" from rod "+str(fro)+" to rod "+str(to))
        self.toh(n-1,aux,to,fro)
        number = 2**n-1
        return number
```

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

```
def permute(s, answer):
    if (len(s) == 0):
        print(answer, end="  ")
        return
    for i in range(len(s)):
        ch = s[i]
        left_substr = s[0:i]
        right_substr = s[i + 1:]
        rest = left_substr + right_substr
        permute(rest, answer + ch)
```

💡 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.

Examples :

Input : abc de Output : 3 There are three consonants b, c and d.
Input : geeksforgeeks portal
Output : 12

```
def isConsonant(ch):
    # To handle lower case
    ch = ch.upper()

    return not (ch == 'A' or ch == 'E' or
                ch == 'I' or ch == 'O' or
                ch == 'U') and ord(ch) >= 65 and ord(ch) <= 90

# To count total number of
# consonants from 0 to n-1
def totalConsonants(string, n):

    if n == 1:
        return isConsonant(string[0])

    return totalConsonants(string, n - 1) + isConsonant(string[n-1])
```