Assignment 10 Solutions

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).

In [24]:
```
def isPowerOfThree(n):
    if n <= 0:
        return False

    while n % 3 == 0:
        n /= 3

    return n == 1
```

In [25]:
```
print(isPowerOfThree(27))  # Output: True
print(isPowerOfThree(0))   # Output: False
```

```
True
False
```

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

In [26]:
```
def lastRemaining(n):
    leftToRight = True
    remaining = n
    step = 1
```

```
    while remaining > 1:
        if leftToRight or remaining % 2 == 1:
            step *= 2

        remaining //= 2
        leftToRight = not leftToRight

    return step
```

In [27]:
```
print(lastRemaining(9))  # Output: 6
```

4

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” }

In [28]:
```
def generateSubsets(s):
    if len(s) == 0:
        return {""}

    first = s[0]
    subset = generateSubsets(s[1:])
    result = set(subset)

    for elem in subset:
        result.add(first + elem)

    return result
```

In [29]:
```
print(generateSubsets("abc"))
# Output: {"", "a", "b", "c", "ab", "ac", "bc", "abc"}

print(generateSubsets("abcd"))
# Output: {"", "a", "ab", "abc", "abcd", "abd", "ac", "acd", "ad", "b", "bc", "bcd", "bd", "c", "cd", "d"}
```

```
{'', 'ac', 'abc', 'a', 'c', 'ab', 'bc', 'b'}
{'', 'ac', 'abcd', 'abc', 'bd', 'a', 'c', 'ad', 'abd', 'ab', 'bc', 'd', 'acd', 'b', 'bcd', 'cd'}
```

Question 4 Given a string calculate length of the string using recursion.

Examples:

Input : str = "abcd"

Output :4

Input : str = "GEEKSFORGEEKS"

Output :13

In [30]:
```
def calculateLength(s):
    if s == "":
        return 0

    return 1 + calculateLength(s[1:])
```

In [31]:
```
print(calculateLength("Hello"))  # Output: 5
print(calculateLength(""))       # Output: 0
print(calculateLength("abc"))    # Output: 3
```

```
5
0
3
```

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

In [32]:
```
def countSubstrings(S):
    count = 0

    for i in range(len(S)):
        for j in range(i, len(S)):
            if S[i] == S[j]:
                count += 1

    return count
```

In [33]:
```
print(countSubstrings("abc"))  # Output: 3
print(countSubstrings("aaa"))  # Output: 6
```

```
3
6
```

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.

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.

In [34]:
```
def towerOfHanoi(N, source, destination, auxiliary):
    if N == 1:
        print("move disk 1 from rod", source, "to rod", destination)
        return 1

    moves = 0
    moves += towerOfHanoi(N-1, source, auxiliary, destination)
    print("move disk", N, "from rod", source, "to rod", destination)
    moves += 1
    moves += towerOfHanoi(N-1, auxiliary, destination, source)

    return moves
```

In [35]:
```
N = 2
totalMoves = towerOfHanoi(N, 1, 3, 2)
print(totalMoves)
```

```
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
```

Question7 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

In [36]:
```
def permute(s):
    if len(s) == 1:
        return [s]

    permutations = []

    for i, ch in enumerate(s):
        remaining_chars = s[:i] + s[i+1:]
        sub_permutations = permute(remaining_chars)

        for perm in sub_permutations:
            permutations.append(ch + perm)

    return permutations
```

In [37]:
```
print(permute("cd"))
# Output: ['cd', 'dc']

print(permute("abb"))
# Output: ['abb', 'bab', 'bba', 'abb', 'bab', 'bba']
```

```
['cd', 'dc']
['abb', 'abb', 'bab', 'bba', 'bab', 'bba']
```

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

In [38]:
```python
def countConsonants(s):
    count = 0

    for ch in s:
        if ch.isalpha() and ch.lower() not in ['a', 'e', 'i', 'o', 'u']:
            count += 1

    return count
```

In [39]:
```python
print(countConsonants("Hello"))  # Output: 3
print(countConsonants("OpenAI")) # Output: 3
print(countConsonants("Python")) # Output: 4
```

```
3
2
5
```

In [ ]: