**Assignment Questions 10**

**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).

Solve:-

class Solution {

public boolean isPowerOfThree(int n) {

if(n==0)

return false;

else

{

while(n%3==0)

n=n/3;

if(n==1)

return true;

}

return 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

Solve:-

class Solution {

public int lastRemaining(int n) {

boolean left = true;

int head = 1;

int step = 1;

int remain = n;

while(remain > 1){

if(left || remain % 2 == 1){

head = head + step;

}

step = step \* 2;

remain = remain / 2;

left = !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” }

Solve:-

import java.util.;

class GFG {

static void powerSet(String str, int index, String curr)

{

int n = str.length();

// base case

if (index == n) {

return;

}

for (int i = index + 1; i < n; i++) {

curr += str.charAt(i);

powerSet(str, i, curr);

curr = curr.substring(0, curr.length() - 1);

}

}

public static void main(String[] args)

{

String str = "abc";

int index = -1;

String curr = "";

powerSet(str, index, curr);

}

}

**Question 4**

Given a string calculate length of the string using recursion.

**Examples:**

Input : str = "abcd"

Output :4

Input : str = "GEEKSFORGEEKS"

Output :13

Solve:-

import java.util.\*;

public class GFG{

/\* Function to calculate length \*/

private static int recLen(String str)

{

if (str.equals(""))

return 0;

else

return recLen(str.substring(1)) + 1;

}

/\* Driver program to test above function \*/

public static void main(String[] args)

{

String str ="GeeksforGeeks";

System.out.println(recLen(str));

}

}

Q**uestion 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

Solve:-

class Solution {

public int numberOfSubstrings(String s) {

int n=s.length();

int i=0;

int j=0;

int ans=0;

Map<Character,Integer>map=new HashMap<>();

while(j<n){

map.put(s.charAt(j),map.getOrDefault(s.charAt(j),0)+1);

// System.out.println(map);

if(!map.containsKey('a') || !map.containsKey('b') || !map.containsKey('c')){

j++;

}

else{

while(map.containsKey('a') && map.containsKey('b') && map.containsKey('c')){

ans+=n-j;

char ch=s.charAt(i);

map.put(ch,map.getOrDefault(ch,0)-1);

if(map.get(ch)==0){

map.remove(ch);

}

i++;

}

j++;

}

}

return ans;

}

}

**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 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:

Solve:-

class Main

{

public static void move(int disks, int source, int auxiliary, int target)

{

if (disks > 0)

{

// move `n-1` discs from source to auxiliary using the target

// as an intermediate pole

move(disks - 1, source, target, auxiliary);

System.out.println("Move disk " + disks + " from " + source + " —> " +

target);

// move `n-1` discs from auxiliary to target using the source

move(disks - 1, auxiliary, source, target);

}

}

// Tower of Hanoi Problem

public static void main(String[] args)

{

int n = 3;

move(n, 1, 2, 3);

}

}

**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

Solve:-

public class PermuteString {

public static String swapString(String a, int i, int j) {

char[] b =a.toCharArray();

char ch;

ch = b[i];

b[i] = b[j];

b[j] = ch;

return String.valueOf(b);

}

public static void main(String[] args)

{

String str = "ABC";

int len = str.length();

System.out.println("All the permutations of the string are: ");

generatePermutation(str, 0, len);

}

public static void generatePermutation(String str, int start, int end)

{

//Prints the permutations

if (start == end-1)

System.out.println(str);

else

{

for (int i = start; i < end; i++)

{

str = swapString(str,start,i);

//Recursively calling function generatePermutation() for rest of the characters

generatePermutation(str,start+1,end);

//Backtracking and swapping the characters again.

str = swapString(str,start,i);

}

}

}

}

**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

class Main {

public static void main(String[] args) {

String line = "This website is aw3som3.";

int vowels = 0, consonants = 0, digits = 0, spaces = 0;

line = line.toLowerCase();

for (int i = 0; i < line.length(); ++i) {

char ch = line.charAt(i);

// check if character is any of a, e, i, o, u

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {

++vowels;

}

// check if character is in between a to z

else if ((ch >= 'a' && ch <= 'z')) {

++consonants;

}

// check if character is in between 0 to 9

else if (ch >= '0' && ch <= '9') {

++digits;

}

// check if character is a white space

else if (ch == ' ') {

++spaces;

}

}

System.out.println("Vowels: " + vowels);

System.out.println("Consonants: " + consonants);

System.out.println("Digits: " + digits);

System.out.println("White spaces: " + spaces);

}

}

There are three consonants b, c and d.

Input : geeksforgeeks portal

Output : 12