Q1class Solution {

public boolean isPowerOfThree(int n) {

if (n <= 0) {

return false;

}

while (n > 1) {

if (n % 3 != 0) {

return false;

}

n /= 3;

}

return n == 1;

}

}

public class Main {

public static void main(String[] args) {

Solution solution = new Solution();

int n = 27;

boolean result = solution.isPowerOfThree(n);

System.out.println(result); // Output: true

}

}

Q2.class Solution {

public int lastRemaining(int n) {

int remaining = n;

int start = 1;

int step = 1;

boolean leftToRight = true;

while (remaining > 1) {

if (leftToRight || remaining % 2 == 1) {

start += step;

}

remaining /= 2;

step \*= 2;

leftToRight = !leftToRight;

}

return start;

}

}

public class Main {

public static void main(String[] args) {

Solution solution = new Solution();

int n = 9;

int result = solution.lastRemaining(n);

System.out.println(result); // Output: 6

}

}

Q3.class Solution {

public void printSubsets(String set) {

printSubsetsHelper(set, "", 0);

}

private void printSubsetsHelper(String set, String currentSubset, int index) {

// Base case: If we have processed all characters in the set

if (index == set.length()) {

System.out.print("\"" + currentSubset + "\" ");

return;

}

// Recursive case 1: Include the current character in the subset

printSubsetsHelper(set, currentSubset + set.charAt(index), index + 1);

// Recursive case 2: Exclude the current character from the subset

printSubsetsHelper(set, currentSubset, index + 1);

}

}

public class Main {

public static void main(String[] args) {

Solution solution = new Solution();

String set = "abc";

solution.printSubsets(set);

}

}

Q4.class Solution {

public int calculateLength(String str) {

return calculateLengthHelper(str, 0);

}

private int calculateLengthHelper(String str, int index) {

// Base case: If we have reached the end of the string

if (index == str.length()) {

return 0;

}

// Recursive case: Add 1 to the length and move to the next character

return 1 + calculateLengthHelper(str, index + 1);

}

}

public class Main {

public static void main(String[] args) {

Solution solution = new Solution();

String str1 = "abcd";

int length1 = solution.calculateLength(str1);

System.out.println("Length of \"" + str1 + "\" is: " + length1);

String str2 = "GEEKSFORGEEKS";

int length2 = solution.calculateLength(str2);

System.out.println("Length of \"" + str2 + "\" is: " + length2);

}

}

Q5.class Solution {

public int countSubstrings(String s) {

int count = 0;

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

count += countPalindromicSubstrings(s, i, i); // odd length palindromes

count += countPalindromicSubstrings(s, i, i + 1); // even length palindromes

}

return count;

}

private int countPalindromicSubstrings(String s, int left, int right) {

int count = 0;

while (left >= 0 && right < s.length() && s.charAt(left) == s.charAt(right)) {

count++;

left--;

right++;

}

return count;

}

}

public class Main {

public static void main(String[] args) {

Solution solution = new Solution();

String s1 = "abcab";

int count1 = solution.countSubstrings(s1);

System.out.println("Count of substrings in \"" + s1 + "\" is: " + count1);

String s2 = "aba";

int count2 = solution.countSubstrings(s2);

System.out.println("Count of substrings in \"" + s2 + "\" is: " + count2);

}

}

Q6.class Solution {

public void towerOfHanoi(int N, int source, int destination, int auxiliary) {

if (N == 1) {

System.out.println("move disk 1 from rod " + source + " to rod " + destination);

return;

}

towerOfHanoi(N - 1, source, auxiliary, destination);

System.out.println("move disk " + N + " from rod " + source + " to rod " + destination);

towerOfHanoi(N - 1, auxiliary, destination, source);

}

public int totalMoves(int N) {

return (int) Math.pow(2, N) - 1;

}

}

public class Main {

public static void main(String[] args) {

Solution solution = new Solution();

int N = 2;

solution.towerOfHanoi(N, 1, 3, 2);

int totalMoves = solution.totalMoves(N);

System.out.println("Total moves: " + totalMoves);

}

}

Q7.import java.util.HashSet;

import java.util.Set;

class Solution {

public void printPermutations(String str) {

Set<String> permutations = new HashSet<>();

generatePermutations(str, "", permutations);

for (String permutation : permutations) {

System.out.println(permutation);

}

}

private void generatePermutations(String str, String current, Set<String> permutations) {

if (str.length() == 0) {

permutations.add(current);

return;

}

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

char ch = str.charAt(i);

String remaining = str.substring(0, i) + str.substring(i + 1);

generatePermutations(remaining, current + ch, permutations);

}

}

}

public class Main {

public static void main(String[] args) {

Solution solution = new Solution();

String str = "abb";

solution.printPermutations(str);

}

}

Q8.public class Main {

public static void main(String[] args) {

String str = "geeksforgeeks portal";

int count = countConsonants(str);

System.out.println("Number of consonants: " + count);

}

public static int countConsonants(String str) {

int count = 0;

str = str.toLowerCase();

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

char ch = str.charAt(i);

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

count++;

}

}

return count;

}

public static boolean isVowel(char ch) {

return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';

}

}