Assignment 10 Solution - Recursion | DSA

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).

Solution Code:

```
package in.ineuron.pptAssignment10;

public class PowerOfThree {
	public static boolean isPowerOfThree(int n) {
		if (n <= 0) {
			return false;
		}

		while (n % 3 == 0) {
			n /= 3;
		}

		return n == 1;
	}

	public static void main(String[] args) {
		int number = 27;
		boolean isPower = isPowerOfThree(number);
		System.out.println(isPower);
	}
}
```

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

Solution Code:

```
package in.ineuron.pptAssignment10;

import java.util.ArrayList;
import java.util.List;

public class LastRemainingNumber {

	public static int lastRemaining(int n) {
		List<Integer> arr = new ArrayList<>();
		for (int i = 1; i <= n; i++) {
			arr.add(i);
		}

		boolean leftToRight = true;
		while (arr.size() > 1) {
			if (leftToRight) {
				for (int i = 0; i < arr.size(); i += 2) {
					arr.remove(i);
				}
			} else {
				for (int i = arr.size() - 1; i >= 0; i -= 2) {
```

```
                                arr.remove(i);
                            }
                        }
                        leftToRight = !leftToRight;
                    }

                    return arr.get(0);
                }

                public static void main(String[] args) {
                    int n = 9;
                    int lastNumber = lastRemaining(n);
                    System.out.println(lastNumber);
                }
            }
```

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 Code:

```
package in.ineuron.pptAssignment10;

public class SubsetPrinter {
	public static void printSubsets(String set) {
		printSubsetsHelper(set, "", 0);
	}

	private static void printSubsetsHelper(String set, String subset, int index) {
		if (index == set.length()) {
			System.out.println(subset);
			return;
		}

		// Exclude the current character
		printSubsetsHelper(set, subset, index + 1);

		// Include the current character
		printSubsetsHelper(set, subset + set.charAt(index), index + 1);
	}

	public static void main(String[] args) {
		String set = "abc";
		printSubsets(set);
	}
}
```

Question 4

Given a string calculate length of the string using recursion.

Examples:

Input : str = "abcd"

Output :4

Input : str = "GEEKSFORGEEKS"

Output :13

Solution Code:

```
package in.ineuron.pptAssignment10;

public class StringLengthCalculator {
	public static int calculateLength(String str) {
		if (str.isEmpty()) {
			return 0;
		} else {
			return 1 + calculateLength(str.substring(1));
		}
	}

	public static void main(String[] args) {
		String str = "abcd";
		int length = calculateLength(str);
		System.out.println(length);
	}
}
```

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

Solution Code:

```
package in.ineuron.pptAssignment10;

public class ContiguousSubstringCounter {
	public static int countContiguousSubstrings(String s) {
		int count = 0;
		int n = s.length();

		// Iterate over all characters
		for (int i = 0; i < n; i++) {
			char startChar = s.charAt(i);
			count++;

			// Count all substrings starting from the current character
			for (int j = i + 1; j < n; j++) {
				char endChar = s.charAt(j);
				if (startChar == endChar) {
					count++;
				}
			}
		}
		return count;
	}

	public static void main(String[] args) {
		String s = "abcab";
		int count = countContiguousSubstrings(s);
		System.out.println(count);
	}
}
```

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.

Solution Code:

```
package in.ineuron.pptAssignment10;

public class TowerOfHanoi {
	public static void solveTowerOfHanoi(int n, int source, int destination, int auxiliary) {
		if (n == 1) {
		System.out.println("Move disk 1 from rod " + source + " to rod " + destination);
			return;
		}

		solveTowerOfHanoi(n - 1, source, auxiliary, destination);
	System.out.println("Move disk " + n + " from rod " + source + " to rod " + destination);
```

```
            solveTowerOfHanoi(n - 1, auxiliary, destination, source);
        }

        public static int calculateTotalMoves(int n) {
            return (int) Math.pow(2, n) - 1;
        }

        public static void main(String[] args) {
            int n = 2;
            solveTowerOfHanoi(n, 1, 3, 2);
            int totalMoves = calculateTotalMoves(n);
            System.out.println(totalMoves);
        }
    }
```

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

Solution Code:

```
package in.ineuron.pptAssignment10;

public class StringPermutations {

	public static void printPermutations(String str) {
		printPermutationsHelper(str, "");
	}

	private static void printPermutationsHelper(String str, String permutation) {
		if (str.isEmpty()) {
			System.out.println(permutation);
			return;
		}

		for (int i = 0; i < str.length(); i++) {
			char currentChar = str.charAt(i);
			String remainingChars = str.substring(0, i) + str.substring(i + 1);
			printPermutationsHelper(remainingChars, permutation + currentChar);
		}
	}

	public static void main(String[] args) {
		String str = "cd";
		printPermutations(str);
	}
}
```

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

Solution Code:

```
package in.ineuron.pptAssignment10;

public class ConsonantCounter {

	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;
	}

	private static boolean isVowel(char ch) {
		return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
	}

	public static void main(String[] args) {
		String str = "abc de";
		int count = countConsonants(str);
		System.out.println(count);
	}
}
```