**Java Assignment 01**

1. **Special Coding questions:**

**Que 01 – Note: a palindrome is a sequence of characters that reads the same forwards and**

**backwards.**

**Ex: Given the following strings...**

**"A man, a plan, a canal: Panama.", return true**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class PalindromeString1{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter the String : ");

String s = sc.nextLine();

String str = s.replaceAll("[^a-zA-Z]","");

str = str.toLowerCase();

String s1 = "";

for(int i = str.length()-1; i >= 0; i--){

s1 = s1 + str.charAt(i);

}

if(str.equals(s1)){

System.out.println("String is palindrome");

}

else{

System.out.println("String is not palindrome");

}

}

}

**==============================================================**

**Que 02 – This question is asked by Google. Given a string, return whether or not it uses**

**capitalization correctly. A string correctly uses capitalization if all letters are capitalized,**

**no letters are capitalized, or only the first letter is capitalized.**

**Ex: Given the following strings...**

**"USA", return true**

**"Calvin", return true**

**"compUter", return false**

**"coding", return true**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class StringCapitalizationCheck2{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter the string : ");

String s = sc.nextLine();

int count = 0;

int count1 = 0;

int init = 0;

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

if(Character.isUpperCase(s.charAt(i))){

count++;

if(Character.isUpperCase(s.charAt(0))){

init++;

}

}

else if(Character.isLowerCase(s.charAt(i))){

count1++;

}

else{

break;

}

}

if(count == s.length() || count1 == s.length() || init == 1){

System.out.println("String is correctly capitalized");

}

else{

System.out.println("String is not correctly capitalized");

}

}

}

**==============================================================**

**Que 03 – This question is asked by Amazon. Given a string representing the sequence of moves a robot vacuum makes, return whether or not it will return to its original position. The string will only contain L, R, U, and D characters, representing left, right, up, and down respectively.**

**Ex: Given the following strings...**

**"LR", return true**

**"URURD", return false**

**"RUULLDRD", return true**

**Ans🡪**

**==============================================================**

**Que 04 – Given two binary strings (strings containing only 1s and 0s) return their sum (also as a binary string).**

**Ans🡪**

public class SumBinaryStrings

{

// This function adds two

// binary strings and return

// result as a third string

static String addBinary(String a, String b)

{

//If the inputs are 0

if(a.charAt(0) == '0' && b.charAt(0) == '0')

{

return "0";

}

// Initialize result

StringBuilder result = new StringBuilder("");

// Initialize digit sum

int s = 0;

// Traverse both strings starting

// from last characters

int i = a.length() - 1, j = b.length() - 1;

while (i >= 0 || j >= 0 || s == 1)

{

// Comput sum of last

// digits and carry

s += ((i >= 0)? a.charAt(i) - '0': 0);

s += ((j >= 0)? b.charAt(j) - '0': 0);

// If current digit sum is

// 1 or 3, add 1 to result

result.append((char)(s % 2 + '0'));

// Compute carry

s /= 2;

// Move to next digits

i--; j--;

}

// Remove leading zeros, if any

int start = result.length()-1;

while(start >=0 && result.charAt(start) == '0')

{

start--;

}

if(start != result.length()-1)

{

result.delete(start+1,result.length());

}

return result.reverse().toString();

}

//Driver code

public static void main(String args[])

{

String a = "1101", b="100";

System.out.print(addBinary(a, b));

}

}

**==============================================================**

**Que 05 – Note: neither binary string will contain leading 0s unless the string itself is 0**

Ex: Given the following binary strings...

"100" + "1", return "101"

"11" + "1", return "100"

"1" + "0", return "1"

**Ans🡪**

5. Note: neither binary string will contain leading 0s unless the string itself is 0

Ex: Given the following binary strings...

"100" + "1", return "101"

"11" + "1", return "100"

"1" + "0", return "1"

6. Given an array of strings, return the longest common prefix that is shared amongst all strings.

import java.lang.\*;

import java.util.\*;

class LongestcommonPrefix6{

static String commonPreUtil(String str1, String str2) {

String result = "";

int n1 = str1.length(), n2 = str2.length();

for (int i = 0, j = 0; i <= n1 - 1 && j <= n2 - 1; i++, j++) {

if (str1.charAt(i) != str2.charAt(j)) {

break;

}

result += str1.charAt(i);

}

return (result);

}

static String commonPre(String arr[], int n) {

String prefix = arr[0];

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

prefix = commonPreUtil(prefix, arr[i]);

}

return (prefix);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the no of the strings in the array : ");

int x = sc.nextInt();

String arr[] = new String[x];

System.out.println("Enter the strings : ");

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

arr[i] = sc.nextLine();

}

int n = arr.length;

String ans = commonPre(arr, n);

System.out.printf("The longest common prefix is : "+ans);

}

}

**==============================================================**

**Que 06 – Given an array of strings, return the longest common prefix that is shared amongst all strings.**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class LongestcommonPrefix6{

static String commonPreUtil(String str1, String str2) {

String result = "";

int n1 = str1.length(), n2 = str2.length();

for (int i = 0, j = 0; i <= n1 - 1 && j <= n2 - 1; i++, j++) {

if (str1.charAt(i) != str2.charAt(j)) {

break;

}

result += str1.charAt(i);

}

return (result);

}

static String commonPre(String arr[], int n) {

String prefix = arr[0];

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

prefix = commonPreUtil(prefix, arr[i]);

}

return (prefix);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the no of the strings in the array : ");

int x = sc.nextInt();

String arr[] = new String[x];

System.out.println("Enter the strings : ");

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

arr[i] = sc.nextLine();

}

int n = arr.length;

String ans = commonPre(arr, n);

System.out.printf("The longest common prefix is : "+ans);

}

}

**==============================================================**

**Que 07 – Note: you may assume all strings only contain lowercase alphabetical characters.**

**Ex: Given the following arrays...**

**["colorado", "color", "cold"], return "col"**

**["a", "b", "c"], return ""**

**["spot", "spotty", "spotted"], return "spot"**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class DuplicateCharacterWithCount7{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter the string : ");

String s = sc.nextLine();

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

int count = 1;

for(int j = i+1; j < s.length(); j++){

if(s.charAt(i) == s.charAt(j)){

count++;

if(count > 1){

System.out.println(s.charAt(i)+" : "+count);

}

}

}

}

}

}

**==============================================================**

**Que 08 – Given a string and the ability to delete at most one character, return whether or not it can form a palindrome.**

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class OccuranceOfCharacter8{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter the String : ");

String s = sc.nextLine();

System.out.println("Enter the character you want to find : ");

char ch = sc.next().charAt(0);

int count = 0;

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

if(s.charAt(i) == ch){

count++;

}

}

System.out.println(ch+" occured in the string "+count+" times.");

}

}

**==============================================================**

**Que 09 – Note: a palindrome is a sequence of characters that reads the same forwards and backwards.**

**Ex: Given the following strings...**

**"abcba", return true**

**"foobof", return true (remove the first 'o', the second 'o', or 'b')**

**"abccab", return false**

**Ans🡪**

import java.util.Scanner;

import java.io.\*;

class Que09\_StringPalindrome {

public static String removeConsecutiveDuplicates(String input)

{

if (input.length() <= 1)

return input;

if (input.charAt(0) == input.charAt(1))

return removeConsecutiveDuplicates(input.substring(1));

else

return input.charAt(0)

+ removeConsecutiveDuplicates(input.substring(1));

}

static void isPalindrome(String s1)

{

String s = removeConsecutiveDuplicates(s1);

int strLength = s.length();

String reverseStr = "";

for (int i = (strLength - 1); i >=0; --i)

{

reverseStr = reverseStr + s.charAt(i);

}

if (s.toLowerCase().equals(reverseStr.toLowerCase())) {

System.out.println(s1 + " is a Palindrome String.");

}

else {

System.out.println(s1 + " is not a Palindrome String.");

}

}

public static void main(String[] args)

{

String str;

// str = "Radar";// reverseStr = "";

str ="abcba";// return true

isPalindrome(str);

str ="foobof";// return true (remove the first 'o', the second 'o', or 'b')

isPalindrome(str);

str ="abccab";// return false

isPalindrome(str);

// System.out.println("Enter the String to check whether it is Palindrome?: ");

// Scanner sc = new Scanner(System.in);

// str = sc.nextLine();

// isPalindrome(str);

}

}

/\***Output - String Palindrom after removing consecutive duplicate characters**

F:\CDAC\CDAC\_Lab\Diwali Homework\Special Coding questions>javac Que09\_StringPalindrome.java

F:\CDAC\CDAC\_Lab\Diwali Homework\Special Coding questions>java Que09\_StringPalindrome

abcba is a Palindrome String.

foobof is a Palindrome String.

abccab is not a Palindrome String.

\*/

**==============================================================**

**Que 10 – Given a string representing your stones and another string representing a list of jewels, return the number of stones that you have that are also jewels.**

**Ex: Given the following jewels and stones...**

**jewels = "abc", stones = "ac", return 2**

**jewels = "Af", stones = "AaaddfFf", return 3**

**jewels = "AYOPD", stones = "ayopd", return 0**

**Ans🡪**

class Que10\_StringCharOccurenace

{

// Method that return count of the given

// character in the string

public static int count(String s, String j)

{

int res = 0;

for(int k=0;k<j.length();k++)

{

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

{

// checking character in string

if (s.charAt(i) == j.charAt(k))

res++;

}

}

return res;

}

// Driver method

public static void main(String args[])

{

String stones, jewels;

stones= "ab";

jewels= "abc";

System.out.println("Jewels which are also stones are:"+(count(stones,jewels)));

stones= "AaaddfFf";

jewels= "Af";

System.out.println("Jewels which are also stones are:"+(count(stones,jewels)));

stones= "AYOPD";

jewels= "ayopd";

System.out.println("Jewels which are also stones are:"+(count(stones,jewels)));

}

}

/\***OUTPUT - FIND COUNT OF CHAR PRESENT IN DESTINATION STRING AND SOURCE STRING**

F:\CDAC\CDAC\_Lab\Diwali Homework\Special Coding questions>java Que10\_StringCharOccurenace

Jewels which are also stones are:2

Jewels which are also stones are:3

Jewels which are also stones are:0

F:\CDAC\CDAC\_Lab\Diwali Homework\Special Coding questions>\*/

**==============================================================**

**Que 11 – Given two strings, s and t, merge the two strings together alternating characters starting with s.**

**Note: If one string is longer than the other, append the remaining characters of the longer string at the end of the merged string.**

test case:

s = "abc", t = "def", return "adbecf".

**Ans🡪**

// Java code to alternatively merge two strings

public class Que11\_StringMerge\_AlternateChar {

// Function for alternatively merging two strings

static String merge(String s1, String s2)

{

// To store the final string

StringBuilder result = new StringBuilder();

// For every index in the strings

for (int i = 0; i < s1.length() || i < s2.length(); i++) {

// First choose the ith character of the

// first string if it exists

if (i < s1.length())

result.append(s1.charAt(i));

// Then choose the ith character of the

// second string if it exists

if (i < s2.length())

result.append(s2.charAt(i));

}

return result.toString();

}

// Driver code

public static void main(String[] args)

{

String s1 = "Sachin";

String s2 = "CDAC";

System.out.println("Input Strings:\n"+s1+"\n"+s2);

System.out.println("Output String:");

System.out.println(merge(s1, s2));

}

}

/\***OUTPUT-**

F:\CDAC\CDAC\_Lab\Diwali Homework\Special Coding questions>java Que11\_StringMerge\_AlternateChar

Input Strings:

Sachin

CDAC

Output String:

SCaDcAhCin

\*/

**==============================================================**

**Que 12 – Given a string, s, return the total number of substring within s that contain the same character.**

**Note: You may assume that s only contains lowercase alphabetical characters.**

**Ex: Given the following string s…**

**s = "aabca", return 6 ("a" appears 3 times, "aa" appears 1 time, "b" appears 1 time, and "c" appears 1 time).**

**Ans🡪**

// Java implementation of the approach

import java.util.Scanner;

import java.io.\*;

class Que12\_Substring\_withSameChar

{

// Function to return the

// number of substrings of

// same characters

static void findNumbers(String s)

{

// Size of the string

int n = s.length();

// Initialize count to 1

int count = 1;

int result = 0;

// Initialize left to 0 and

// right to 1 to traverse the

// string

int left = 0;

int right = 1;

while (right < n)

{

// Checking if consecutive

// characters are same and

// increment the count

if (s.charAt(left) == s.charAt(right))

{

count++;

}

// When we encounter a

// different characters

else

{

// Increment the result

result += count \* (count + 1) / 2; // when same char not pesent then it increment with 1 and when same char founds it increments with permutation of that no of characters.

//like if same char are 3 times then it increment with 6, bbb,bb,bb,b,b,b times; when 2 increments with 3 times aa,a,a

// To repeat the whole

// process set left equals

// right and count variable to 1

left = right;

count = 1;

}

right++;

}

// Store the final

// value of result

result += count \* (count + 1) / 2; // this increment for last iteration when string ends

System.out.println("No. of Possible substrings are:"+result);

}

// Driver code

public static void main (String[] args)

{

String s;// = new String();

// s = "aabca";

Scanner sc = new Scanner(System.in);

System.out.println("Enter the string:");

s = sc.nextLine();

findNumbers(s);

}

}

/\***OUTPUT - NO. OF SUBSTRING ONLY LOWERCASE**

F:\CDAC\CDAC\_Lab\Diwali Homework\Special Coding questions>java Que12\_Substring\_withSameChar

Enter the string:

aabca

No. of Possible substrings are:6

OUTPUT - NO. OF SUBSTRING ONLY MIXED

F:\CDAC\CDAC\_Lab\Diwali Homework\Special Coding questions>java Que12\_Substring\_withSameChar

Enter the string:

Aabca

No. of Possible substrings are:5

F:\CDAC\CDAC\_Lab\Diwali Homework\Special Coding questions>

\*/

**==============================================================**

**Que 13 – Given an encoded string, s, return its decoded representation. The string s will be encoded as follows N[letters], meaning that the letters should be repeated N times in the decoded representation. Note: You may assume s always encoded correctly (i.e. correct formatting of brackets, only positive values outside the brackets, and always lowercase alphabetical characters inside the brackets).**

Ex: Given the following string s… s = "3[a]2[b]1[c]", return "aaabbc" ("a" is repeated 3 times, "b" is repeated 2 times, and "c" is repeated 1 time).

**Ans🡪**

import java.lang.\*;

import java.util.\*;

class EncodedDecodedString13{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.println("Enter the encoded string : ");

String s = sc.nextLine();

String ch = s.replaceAll("[^a-zA-Z]", "");

String num = s.replaceAll("[^0-9]", "");

int x[] = new int[num.length()];

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

x[i] = Character.getNumericValue(num.charAt(i));

}

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

int k = x[i];

for(int j = 0; j < k; j++){

System.out.print(ch.charAt(i));

}

}

}

}

**==============================================================**

**Que 14 – You are given a list of strings, times, where each string represent a timestamp of a twenty-four hour clock (i.e. hours and minutes - “HH:MM”). Return the minimum difference, in minutes, between any two of the timestamps in the list. Ex: Given the following times… times = ["01:00", "01:10"], return 10 (i.e. there are 10 minutes between the two times). Ex: Given the following times… times = ["00:00", "12:23", "05:50", "23:12"], return 48.**

**Ans🡪**

Did not got its solution

**==============================================================**

**Que 15 – Given a string, s, return the length of the longest substring that contains every vowel occurring an even number of times.**

Note: You may assume s only contains lowercase alphabetical characters and the vowels you must account for are a, e, i, o, and u.

Ex: Given the following string s…

s = "aeiouaeioua", return 10 (the last 'a' cannot count).

Ex: Given the following string s…

s = "bbb", return 3 (all vowels occur an even number of times, i.e. zero times each).

**Ans🡪**

**==============================================================**

**Que 16 – You are given a list of words and asked to find the longest chain. Two words (or more) form a chain if a single letter can be added anywhere in a word, s, to form a word, t (where s and t are both words within the list of words you’re given). Return the length of the longest chain you can form.**

Ex: Given the following words…

words = ["a", "ab", "abc"], return 3 ("a" can be transformed to "ab" by adding a "b" and "ab" can be transformed by adding a "c" giving a chain length of 3).

Ex: Given the following words…

words = ["a", "abc"], return 1 (both "a" or "abc" form their own chains of size one, they cannot be added together).

**Ans🡪**

**==============================================================**

**Que 17 –You are given two string arrays, queries and words. For any string, s, f(s) is equal to the number of times the smallest lexicographical characters occurs in s. For each query, queries[i] count the total number of words where f(queries[i]) < f(word) and return the answer as an array.**

Note: Both queries and words will only contain lowercase alphabetical characters and contain at most ten strings each.

Ex: Given the following queries and words…

queries = ["abc"], words = ["def"], return 0 ('a' and 'd' both occur once so f("abc") and f("def") are equal).

Ex: Given the following queries and words…

queries = ["abc"], words = ["ddef", "xxyz"], return 2 ('a' appears once and 'd' and 'x' appear twice so f("abc") is less than both f("ddef") and f("xxyz")).

**Ans🡪**

**==============================================================**