STRING DATA STRUCTURE

Content

2.1. Anagram

```
boolean areAnagram(String str1, String str2)
{
        // If both strings are of different length. Return false
        if (str1.length() != str2.length() )
                return false;

        // Create a count array and initialize all values as 0
        int count[] = new int[256];
        int i;
        // For each character in input strings, increment count in
        // the corresponding count array
        for (i = 0; i < str1.length() ; i++) {
                count[str1.charAt ( i ) ]++;
                count[str2.charAt ( i ) ]--;
        }

        // See if there is any non-zero value in count array
        for (i = 0; i < 256 ; i++)
                if (count[i] != 0)
                        return false;
        return true;
}
```

2.2 Reverse words in a given string

```
public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for ( int i = 0 ; i < t ; + + i ){
                String str = sc.next();
                ArrayList <String> list = new ArrayList<>();
                int l = str.length();
                int k = 0;
                String word = "";
                while ( k < l ){
                        char c = str.charAt ( k++);
                        if( c == '.' ){
                                list.add(word);
                                word = "";
                        }else{
                                word += c;
                        }
                }
                list.add(word);
                for ( int j = list.size () - 1 ; j >= 0 ; j--){
                        System.out.print(list.get(j));
                        if(j!=0){
                                System.out.print(".");
                        }
                }
                System.out.println();
        }
}
```

2.3. Remove duplicates

```
public static void main (String[] args) {
        Scanner sc = new Scanner ( System.in);
        int t = sc.nextInt();
        sc.nextLine();
        for ( int tt = 0 ; tt < t ; tt++){
                String str = sc.nextLine();
                int l = str.length();
                HashSet<Character> set = new HashSet<>();
                for(int i=0;i<l;i++){
                        char c = str.charAt(i);
                        if(!set.contains(c)){
                                System.out.print(c);
                                set.add(c);
                        }
                }
                System.out.println();
        }
}
```

2.4. Longest distinct character in a string

```
static int nRCS(String str){
        int curr_len = 1;
        int max_len = 1;
        int prev_index = 0;
        int visited[] = new int[256];
        for ( int i=0 ; i<256 ; i++ ) {
                visited[i] = -1;
        }
        visited[ str.charAt ( prev_index ) ] = 0;
        for( int i = 1 ; i < str.length ( ) ; i ++){
                prev_index = visited [ str.charAt(i) ];
                if(prev_index == -1 || i – curr_len > prev_index ) {
                        curr_len++;
                }else{
                        if ( curr_len > max_len){
                                max_len = curr_len;
                        }
                        curr_len =  i - prev_index;
                }
                visited[str.charAt(i)] = i;
        }
        if(curr_len>max_len){
                max_len = curr_len;
        }
        return max_len;
}
public static void main (String[] args) {
         Scanner sc = new Scanner(System.in);
         int test = sc.nextInt();
         for ( int t = 0 ; t < test ; t ++){
                String str = sc.next();
                System.out.println( nRCS ( str ) );
         }
}
```

2.5. Check if string is rotated by two places

```
public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int test = sc.nextInt();
        for ( int t = 0 ; t < test ; t ++){
                String str1 = sc.next();
                String str2 = sc.next();
                int m = str1.length();
                int n = str2.length();
                int res = 0;
                if(m == n){
                        if ( m > 2 ) {
                                String s = str2.substring ( n – 2 , n ) ;
                                s += str2.substring ( 0 , n – 2 ) ;
                                if ( s.equals ( str1 ) ) {
                                        res = 1;
                                }else{
                                        s = str2.substring ( 2 , n ) ;
                                        s += str2.substring ( 0 , 2 ) ;
                                        if ( s.equals ( str1 ) )
                                                res = 1;
                                }
                        }else{
                                if ( str1.equals ( str2 ) )
                                        res = 1;
                        }
                }
                System.out.println (res) ;
        }
}
```

2.6. Roman number to integer

```
static int value(char c)
{
        int res = -1;
        switch(c){
              case 'I':
                        res = 1;
                        break;
              case 'V':
                        res = 5;
                        break;
              case 'X':
                        res = 10;
                         break;
              case 'L':
                        res = 50;
                        break;
              case 'C':
                        res = 100;
                        break;
              case 'D':
                        res = 500;
                        break;
              case 'M':
                         res = 1000;
        }
        return res;
}
static int romanToInt(String str){
        int res = 0;
        for(int i=0;i<str.length();i++){
                int s1 = value(str.charAt(i));
                if(i+1<str.length()){
                        int s2 = value(str.charAt(i+1));
                        if(s1>=s2){
                                res = res + s1;
                        }else{
                                res = res + s2 - s1;
                                i++;
                        }
```

```
                }else{
                        res = res + s1;
                        i++;
                }
        }
        return res;
}
public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int test = sc.nextInt();
        for ( int t = 0 ; t < test ; t ++){
                String str = sc.next();
                System.out.println(romanToInt(str));
        }
}
```

2.7. Implement strstr

//the function returns the position where the target string matches

//the string str

```
int strstr(String str, String target)
{
        int m = str.length();
        int n = target.length();
        if(m>=n){
                for(int i=0;i<m-n+1;i++){
                        String temp = str.substring(i,i+n);
                        if(temp.equals(target)){
                                return i;
                        }
                }
       }
       return -1;
}
```

2.8. Implement Atoi

```
//The function takes a string(str) as argument and converts it to an integer
//and returns it.

int atoi(String str)
{
    int res = 0,flag=0;
    int n = str.length();
    for(int i=0;i<n;i++){
            char c = str.charAt(i);
            int s = c;
            if(c=='-'){
                    flag = 1;
            }else if(s<48||s>57){
                    return -1;
            }else
                    res = res * 10 + Character.getNumericValue(c);
    }
    if(flag==1){
            res = res*-1;
    }
    return res;
}
```

2.9. Longest common prefix in an array

```
static String longestPrefix(String ar[],int n){
        if(n==0)
                return "";
        if(n==1)
                return ar[0];
        Arrays.sort(ar);
        int end = Math.min ( ar [ 0 ].length(), ar [ n – 1 ].length());
        int i = 0;
        while ( i < end && ar [ 0 ].charAt ( i ) == ar [ n – 1 ].charAt ( i ) ) {
                i++;
        }
        return ar[0].substring(0,i);
}

public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int test = sc.nextInt();
        for(int t = 0; t<test;t++){
                int n = sc.nextInt();
                String ar[] = new String[n];
                for(int i=0;i<n;i++){
                        ar[i] = sc.next();
                }
                String res = longestPrefix(ar,n);
                if(res.length()==0){
                        System.out.println(-1);
                }else{
                        System.out.println(res);
                }
        }
}
```

2.10. Permutations of a given string

```
class GFG {
	static void permute ( String str, int l, int r, ArrayList<String> ans){
		if ( l == r ) {
			ans.add ( str ) ;
		}else{
			for ( int i = l ;i <= r ; i++){
				str = swap ( str, l, i);
				permute(str,l+1,r,ans);
				str = swap ( str, l, i);
			}
		}
	}
	static String swap(String str, int i, int j ) {
		char arr[] = str.toCharArray();
		char temp = arr[i];
		arr[i] = arr[j];
		arr[j] = temp;
		return String.valueOf(arr);
	}
	public static void main (String[] args) {
		Scanner sc = new Scanner(System.in);
		int test = sc.nextInt();
		for(int t=0;t<test;t++){
			String str = sc.next();
			int m = str.length();
			ArrayList<String> ans = new ArrayList<>();
			permute(str, 0, m – 1, ans);
			Collections.sort(ans);
			for(int i=0;i<ans.size();i++){
				System.out.print(ans.get(i)+" ");
			}
			System.out.println();
		}
	}
}
```

2.11. Recursively remove all adjacent duplicates

```
class LastRemoved{
        char last_removed;
        int flag;
        public LastRemoved(){}
}
class GFG {

        static LastRemoved last;
        static String remove(String str){

                // If length of string is 1 or 0
                if (str.length() == 0 || str.length() == 1)
                        return str;

                // Remove leftmost same characters and recur for remaining
                // string
                if (str.charAt(0) == str.charAt(1)){
                        last.last_removed = str.charAt(0);
                        while (str.length() > 1 && str.charAt(0) == str.charAt(1))
                                str = str.substring(1, str.length());
                        str = str.substring(1, str.length());
                        return remove(str);
                }

                // At this point, the first character is definiotely different
                // from its adjacent. Ignore first character and recursively
                // remove characters from remaining string
                String rem_str = remove(str.substring(1,str.length()));

                // If remaining string becomes empty and last removed character
                // is same as first character of original string. This is needed
                // for a string like "acbbcddc"
                if (last.flag == 1 && last.last_removed == str.charAt(0)){
                        return rem_str;
                }

                // Check if the first character of the rem_string matches with
                // the first character of the original string
                if (rem_str.length() != 0 && rem_str.charAt(0) == str.charAt(0)){
```

```
                last.last_removed = str.charAt(0);
                last.flag = 1;
                return rem_str.substring(1,rem_str.length()); // Remove first
character
        }else if(rem_str.length() != 0){
                last.flag = 0;
        }

        // If the two first characters of str and rem_str don't match,
        // append first character of str before the first character of
        // rem_str

        return (str.charAt(0) + rem_str);
    }
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int test = sc.nextInt();
        last = new LastRemoved();
        for(int t = 0;t<test;t++){
                String str = sc.next();
                last.last_removed = '\0';
                last.flag = 0;
                System.out.println(remove(str));
        }
    }
}
```

2.12. Form a palindrome

```
static int lcs ( String str1, String str2, int m, int n ){
        int dp[][] = new int[m+1][n+1];
        for(int i=0;i<=m;i++){
                for(int j=0;j<=n;j++){
                        if(i==0 || j==0){
                                dp[i][j] = 0;
                        }else if(str1.charAt(i-1)==str2.charAt(j-1)){
                                dp[i][j] = dp[i-1][j-1]+1;
                        }else{
                                dp[i][j] = Math.max(dp[i-1][j],dp[i][j-1]);
                        }
                }
        }
        return dp[m][n];
}
static int makePalindrome(String str){
        StringBuffer s = new StringBuffer(str);
        s.reverse();
        String rev = s.toString();
        int longestCommonSubsequence = lcs(str,rev,str.length(),rev.length());
        return ( str.length() - longestCommonSubsequence);
}
public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for(int i=0;i<t;i++){
                String str = sc.next();
                System.out.println(makePalindrome(str));
        }
}
```

2.13. Longest common substring

```
static int lcs ( String str1, String str2, int m, int n , int count ) {
        if(m == 0 || n == 0)
                return count;
        if ( str1.charAt ( m – 1 )  == str2.charAt (n – 1 ) ) {
                count = lcs (str1, str2, m-1, n-1, count+1 );
        }
        count = Math.max(count, Math.max ( lcs(str1, str2, m, n-1, 0), lcs (str1,
str2, m – 1, n ,0 ) ) ) ;
        return count;
}
public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int test = sc.nextInt();
        for ( int t = 0 ; t < test ; t++ ){
                int m = sc.nextInt();
                int n = sc.nextInt();
                String str1 = sc.next();
                String str2 = sc.next();
                System.out.println(lcs(str1,str2,m,n,0));
        }
}
```

DP:

```
static int lcs(char str1[],char str2[],int m,int n){
        int dp[][] = new int[m+1][n+1];
        int result=0;
        for ( int i = 0 ; i <= m ; i ++){
                for ( int j =0 ; j <= n ; j++){
                        if ( i ==0 || j == 0 ) {
                                dp[i][j] = 0;
                        }else if(str1[i-1]==str2[j-1]){
                                dp[i][j] = dp[i-1][j-1] +1 ;
                                result = Math.max(result,dp[i][j]);
                        }else{
                                dp[i][j] = 0;
                        }
                }
        }
        return result;
```

```
}

public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int test = sc.nextInt();
        for ( int t = 0 ; t < test ; t++){
                int m = sc.nextInt();
                int n = sc.nextInt();
                String str1 = sc.next();
                String str2 = sc.next();
                System.out.println(lcs(str1.toCharArray(),str2.toCharArray(),m,n));
        }
}
```