

## 1) Playing with Number

```
static int[] shiftArray(int input1, int[] input2, int input3)
{
    int ar[]=new int[input2.length];
    int j=0;
    for(int i=input3;i<input2.length;i++)
    {
        ar[j++]=input2[i];
    }
    for(int i=0; i<input3;i++)
    {
        ar[j++]=input2[i];
    }
    return ar;
}
```

Test cases

```
input1=7;
input2[]={1,2,3,4,5,6,7};
input3=2;
```

output:

```
3 4 5 6 7 1 2
```

## 2) Array Sort(nick checks)

```
private static int getCount(int input1, int[] input2)
{
    int ar[]=new int[input2.length];
    for(int i =0;i<input2.length;i++)
    {
        ar[i]=input2[i];
    }
    Arrays.sort(ar);
    int num=ar[0];
    for(int i=0;i<input2.length;i++)
    {
        if(ar[i]!=num)
            return 0;
        num++;
    }
    return 1;
}
```

Test Case:

input1:6

input2[]: {3,7,2,5,4,6}

output:

1

### 3) Fancy Occurrence

```
private static String fancyOcc(String input1, String input2)
{
    String s="";
    char com=input2.charAt(0);
    for(int i= 0; i<input1.length();i++)
    {
        char ch=input1.charAt(i);
        if(ch!=com)
            s=s+ch;
    }
    return s;
}
```

Test Case:

input1:Welcome to metti

input2:i

output:

Welcome to mett

## 4) String Within String

```
private static String isPermutation(String input1,String input2)
{
    input1=input1.toLowerCase();
    input2=input2.toLowerCase();
    int arr1[] = new int[26];
    int arr2[] = new int[26];
    for(int i=0; i<input1.length();i++)
    {
        char ch = input1.charAt(i);
        arr1[ch-97]++;
    }
    for(int i=0;i<input2.length();i++)
    {
        char ch1 = input2.charAt(i);
        arr2[ch1-97]++;
        if(arr1[ch1-97]<arr2[ch1-97])
            return "no";
    }
    return "yes";
}
```

Test Case:

input1: abab

input2: ab

output:

yes

## 5) Palindrome

```
private static int isPalindrome(String input1)
{
    int j = input1.length();
    for(int i=0;i<s.length()/2;i++)
    {
        char ch= input1.charAt(i);
        if(ch!= input1.charAt(j-1))
            return 0;
        j--;
    }
    return 1;
}
```

Test Case:

input1: level

output:

1

## 6) Max occurring character

```
private static String maxOccurance(String input1)
{
    char[] c=new char[26];
    String s="";
    for(int i=0;i<input1.length();i++)
    {
        char ch= input1.charAt(i);
        c[ch-97]++;
    }
    int max=0;
    for(int i=1;i<26;i++)
    {
        if(c[max]<c[i])
            max=i;
    }
    for(int i=0;i<26;i++)
    {
        if(c[i]==c[max]&&i!=max)
            return "0";
    }
}
```

```
        return s+(char)(max+97);
    }
```

Test cases:

input1 : abcdd

output : d

## 7) Number Sum

```
private static int sum(int input1, int[] input2)
{
    int max=input2[0];
    int min=input2[0];
    for(int i=1;i<input2.length;i++)
    {
        if(max<input2[i])
            max=input2[i];
        if(min>input2[i])
            min=input2[i];
    }
    return max+min;
}
```

Test cases:

input 1: 7

input 2: { 7,2,6,15,54,10,23}

output : 56

## 8) Anagram

```
private static String isAnagram (String input1,String input2)
{
    int n1 = input1.length();
    int n2 = input2.length();
    if (n1 != n2)
        return "no";
    input1=input1.toUpperCase();
    input2=input2.toUpperCase();
    int a[]= new int[26];
    int a2[]= new int[26];
    for(int i=0;i<n1;i++)
    {
        char ch= input1.charAt(i);
        a[ch-65]++;
    }
    for(int i=0;i<n2;i++)
    {
        char ch= input2.charAt(i);
        a2[ch-65]++;
    }

    for (int i = 0; i < 26; i++)
        if (a[i] != a2[i])
            return "no";
    return "yes";
}
```

Test cases:

```
input1:build
input : dubli
output : yes
```

## 9) Coin Counts

```
private static int getcount(int input1)
{
    int coin=0;
    while(input1!=0)
    {
        coin=coin+(input1*input1);
        input1--;
    }
    return coin;
}
```

Test cases:

input1 : 2

output : 5

## 10) Electro Static

```
private static int electroStatic(int[] input1, String input2, int
input3)
{
    int sum=0;
    for(int i = 0;i<input3;i++)
    {
        char ch = input2.charAt(i);
        switch(ch)
        {
            case 'P':
                sum=sum+input1[i];
                break;
            case 'N':
                sum=sum-input1[i];
                break;
        }
    }
    return sum*100;
}
```

Test cases:

input1 : {4,3,5}

input2 : PNP

input3 : 3

output : 600

## 11) Reverse Array

```
public static int[] reverseArray(int [] input1,int input2) {  
    int [] ars=new int[input2];  
    int k=0;  
    for(int i=input2-1;i>=0;i--)  
    {  
        ars[k]=input1[i];  
        k++;  
    }  
    return ars;  
}
```

Test cases:

input1 : {1 , 2 , 3 ,4, 5}

input2 : 5

output : 5 4 3 2 1



## 12) Modular equation

```
private static int isModule(int input1, int input2, int input3)
{
    int rs=pow(input1,input2);
    return rs%input3;
}

private static int pow(int input1, int input2) {
    int fact=1;
    while(input2>0)
    {
        fact=fact * input1;
        input2--;
    }
    return fact;
}
```

Test cases :

input1 : 2

input2 : 10

input3 : 1025

output : 1024

### 13) Remove duplicates from String

```
static String removeDuplicates(String input1)
{
    String rs="";
    char ch[]=input1.toCharArray();
    for(int i=0;i<ch.length;i++)
    {
        if(rs.indexOf(ch[i])!=-1)
        rs=rs+ch[i];
    }
    return rs;
}
```

Test cases

input1 : geeksforgeeks

output : geksfors

### 14) Push zero to end

```
static void pushZerosToEnd(int input1[], int input2)
{
    int count = 0;
    for (int i = 0; i < input2; i++)
        if (input1 [i] != 0)
            input1 [count++] = input1 [i];
    while (count < input2)
        input1 [count++] = 0;
}
```

Test cases:

input1 : {1,0,2,5,0,6,0,0,9}

input2 : 9

output : 1 2 5 6 9 0 0 0 0

## 15) Cuckoo

```
static void cuckoo(int n)
{
    int cuckoo1 = 0, cuckoo2 = 1;
    int counter = 0;
    int t=0;

    // Iterate till counter is n
    while (counter < n) {

        // Print the number

        t=cuckoo1;
        // Swap
        int cuckoo3 = cuckoo2 + 2*cuckoo1 +3*1;
        cuckoo1 = cuckoo2;
        cuckoo2 = cuckoo3;
        counter = counter + 1;
    }
    System.out.print(t + " ");
}
```

Test cases

Input1 : 2

Output: 1

## 16) Remove Duplicates from array

```
public static int removeDuplicate(int input1[], int input2)
{
    if(input2==0 || input2==1)
        return input2;

    int [] t=new int[input2];
    int j=0;
    for(int i=0;i<input2-1; i++)
    {
        if(input1[i]!=input1[i+1]) {
            t[j++]=input1[i];
        }
    }
    t[j++]=input1[input2-1];
    for(int i=0;i<j; i++)
    {
        input1[i]= t[i];
    }
    return j;
}
```

Test cases:

Input1: {1,2,2,11,11,11,15,6}

Input2 : 8

Output : 1 2 11 15 6

## 17) Missing Braces

```
static String areBracketsBalanced(String expr)
{
    Deque<Character> stack
        = new ArrayDeque<Character>();
    for (int i = 0; i < expr.length(); i++)
    {
        char x = expr.charAt(i);

        if (x == '(' || x == '[' || x == '{')
        {
            stack.push(x);
            continue;
        }
        if (stack.isEmpty())
            return "error";
        char check;
        switch (x) {
            case ')':
                check = stack.pop();
                if (check == '{' || check == '[')
                    return "error";
                break;
            case '}':
                check = stack.pop();
                if (check == '(' || check == '[')
                    return "error";
                break;
            case ']':
                check = stack.pop();
                if (check == '(' || check == '{')
                    return "error";
                break;
        }
        if (stack.isEmpty())
            return "correct";
        else
            return "error";
    }
}
```