Question-1

Problem Statement:

The binary number system only uses two digits, 0 and 1. Any string that represents a number in the binary number system can be called a binary string. You are required to implement the following function:

```
int OperationsBinaryString(char *str);
```

The function accepts a string 'str' as its argument. The string 'str' consists of binary digits separated with an alphabet as follows:

'A' denotes AND operation
'B' denotes OR operation
'C' denotes XOR operation

You are required to calculate the result of the string 'str', scanning the string left to right, taking one operation at a time, and return the same.

Question-1

Problem Statement:

Note:

No order of priorities of operations is required.

Length of 'str' is odd

If 'str' is NULL or None(in case of python), return -1

Example

Input:

ICOCICIAOBI

Output:

1

Explanation:

The alphabet in 'str' when expanded becomes "1 XOR 0 XOR 1 XOR 1 AND 0 OR 1", the result of the expression becomes 1, hence 1 is returned.

Solution

```
import java.util.*;
import java.lang.*;
class Main
{
    public static int OperationsBinaryString(
char[] str)
    {
        int len=str.length;
        int ans= str[0]-'0';
        for(int i=1;i<len-1;i+=2)
        {
            int j=i+1;
            if(str[i]=='A')
            {
                ans = ans & (str[j]-'0');
            }
            else if(str[i]=='B')
            {
                ans = ans | (str[j]-'0');
            }
else if(str[i]=='C')
            {
                ans = ans ^ (str[j]-'0');
            }
        }
    return ans;
 }
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        char[] str=s.toCharArray();
        System.out.printf("%d",OperationsBina
ryString(str));
    }
}
```

Question-2

Problem Statement:

A palindrome is a sequence of characters that has the property of reading the same in either direction. You are given a function,
char* ConvertToPalindrome(char*str);
The function accepts a string ‘str’. Implement the function to find and return the minimum characters required to append at the end of string ‘str’ to make it palindrome.
Assumption:
String will contain only lower case English alphabets.
Length of string is greater than equal to 1.
Note:
If string is already palindrome then return “NULL”.
You have to find the minimum characters required to append at the end of string to make it palindrome

Question-2

Problem Statement:

Example:

Input:

abcdc

Output:

ba

Explanation:

If we append ‘ba’ at the end of the string ‘abcdc’ it becomes ‘abcdcba’(i.e A palindrome string)

solution

```
import java.util.*;
class Main
{
    public static char[] append(String s)
    {
        int l =s.length();
        int j=l-1,count=0;
        for(int i =0;i<l;i++)
        {
            while(s.charAt(i)!=s.charAt(j))
            {
                i++;
                count++;
            }
            j--;
        }
 StringBuffer str=new StringBuffer(s.substrin
g(0,count));
        String st=str.toString();
        char[] pali = st.toCharArray();
        return pali;
    }
public static void main(String[] args)
    {
        Scanner sc =new Scanner(System.in);
        String s=sc.nextLine();
        char[] ch=append(s);
        for (int i = ch.length-1; i >= 0 ; i-
-)
        {
            System.out.print(ch[i]);
        }
    }
}
```

Question-3

Problem Statement:

Cubic sum

You are given a function:

Int isCubicSumExist(long long int A[], int N);

The function accepts an array ‘A’ of size ‘N’ implement the function to return the count of good integers in array ‘A’

An integer Z is said to be good if and only if there exist two integers x and y such that x3 + y3 = z

Question-3

Problem Statement:

Example
Input:
N :3
A : [35,9,1]
Output:
2
Explanation:
35 is a good integer, there exist an answer with X=2,Y=3(23+3 3 = 8+27 = 35)
9 is a good integer,there exist an answer with X=1,Y=2(13 + 2 3   = 9)
1 is not a good integer, so total 2 integers are good in the given array A
The custom input format for the above case
3
35 9 1
(the first line represents ‘N’ the second line represents the elements of the array ‘A’)

solution

```
import java.util.*;
class Main
{
   public static boolean sumOfTwoCubes(int n)
    {
        int lo = 1, hi = (int)Math.cbrt(n);
        while (lo <= hi)
        {
            int curr = (lo * lo * lo + hi * hi * hi);
            if (curr == n)
                return true;
            if (curr < n)
                lo++;
            else
                hi--;
        }
        return false;
    }
```

```
public static int cubicSum(int n, int arr[])
    {   int count = 0;
        for(int i = 0; i < n; i++)
        {
            if (sumOfTwoCubes(arr[i]))
            {
                count++;
            } }
        return count;
    }
    public static void main (String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int arr[] = new int[N];
        for(int i = 0; i < N; i++)
            arr[i] = sc.nextInt();
        System.out.println(cubicSum(N, arr));
    }}
```

Question-4

Problem Statement:

Implement the following function:
Int PlayList(int airtime, int songs[], int n);

The function accepts a positive integer ‘airTime’ and a positive integer array ‘songs’ of size ‘n’ as its argument. ‘ songs’ consists of length of songs (in minutes). A radio jockey has to playlists of combination of exactly thre songs such that the total length of playlists is equal to ‘airtime’ (in minutes). Implement the function to find the count of playlist that can be find and return the same.
Assumption: ‘songs’ consists of unique elements
Note: Return -1 if ‘songs’ is null(None,in case of python) or n<3

Question-4

Problem Statement:

```
Example:
Input:
airTime : 40
Songs : 7 14 21 19 17 2 29 5
Output:
3
Explanation:
Playlists formed are
{14,21,5} = 14 + 21+ 5 = 40
{7,14,19} = 7 + 14 + 19 = 40
{21,17,2} = 21 + 17 + 2 = 40
Since, 3 playlists can be formed thus, output is 3
The custom input format for the above case:
40
8
7 14 21 19 17 2 29 5
(the first line represents ‘airTime’ the second line represents the size of ‘songs’,
The third line represents the element of ‘songs’)
```

Question-4

Problem Statement:

Sample Input:
airTime: 21
songs : 10 7 9 5 2
2
The custom input for the above case:
21

5
10 7 9 5 2
(the first line represents ‘airTime’,the second line represents the size of ‘songs’,
The third line represents the element of ‘songs’)
Instructions:
This is a template based question,DO NOT write the “main” function
Your code is judged by an automated system,do not write any additional welcome /greeting messages
“Save and Test” only checks for basic test cases, more rigorous cases will be used to judge your code while scanning
Additional score will be given for writing optimized code both in terms of memory and execution time

Solution

```
import java.util.*;
class Main
{
    public static int playList(int airTime, i
nt songs[], int n )
    {
        int count=0;
        for (int i = 0; i < n - 2; i++)
        {
            HashSet<Integer> s = new HashSet<
Integer>();
            int curr_sum = airTime -
 songs[i];
            for (int j = i + 1; j < n; j++)
            {
                if (s.contains(curr_sum -
 songs[j]))
{count++;
                }
                s.add(songs[j]);
            }  }
if(count>0)
            return count;
        return -1;
    }
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int air_time=sc.nextInt();
        int n=sc.nextInt();
        int[] songs =new int[n];
        for(int i=0;i<n;i++)
            songs[i]=sc.nextInt();
        System.out.println(playList(air_time,
 songs, n));
    }
}
```

Question-5

Problem Statement:

Find Smallest Character

You are given a function:

def SmallestCharacter(s):

The function accepts a string ‘s’. Implement the function to find the smallest English character which ios not present in the given string ‘s’ and return the same.

Question-5

Problem Statement:

Example :

Input :

aidubudxd

Output :

c

Explanation :

Input string contains a and b. So now the smallest character that is not present in the string is c.

The custom input format for the above case:

aidubndxd

(The line represents a string ‘s’)

Sample Input

```
 bbbb
```

Sample Output

```
a
```

Solution

```
import java.util.*;
class Main
{
    public static char smallestCharacter(Stri
ng str)
    {
        int freq[] = new int[26];
        int len = str.length();

        for(int i = 0; i <len; i++)
        {
            freq[str.charAt(i) - 'a']++;
        }
        for(int i = 0; i < 26; i++)
```

```
  {
            if(freq[i] == 0)
            {
                int num = 'a' + i;
                char ch = (char) (num);
                return ch;
            }

 }
        return 'a';
    }
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String str = sc.next();
   System.out.print(smallestCharacter(str));
    }
 }
```

Question-6

Problem Statement:

Find the word average
Implement the following function:
Static float Average(String str){}
The function accepts a string 'str' of length 'len' as its arugment. Implement the function to calculate the word average and return the same. Word Average is calculated by finding the average of the ASCII values of all of the letters in a word.
Note:

- 'str' is not null
- Input string will contain only lower case English alphabets
- The ASCII value of lower case 'a' is 97 while that of 'z' is 122
- Do not round off your results, it will be automatically rounded off up to 2 decimal places and then displayed

Question-6

Problem Statement:

Example :

Input :

Str: source

Output :

109.50

Explanation :

Char	value
S	115
o	111
u	117
r	114
c	99
e	101

Question-6

Problem Statement:

Word Average =(115+111+117+114+99+101)/6=657/6=109.50

Thus Output is 109.50

The custom input format for the above case :

6 source

(The first line represents ‘len’, the second line represent the string ‘str’)

Sample Input

Str: asp

Sample Output

108.00

The custom input format for the above case:

3asp

(The first line represents ‘len’, the second line represents the string ‘str’).

Solution

```
import java.util.*;
class Main
{
    public static float average(String str)
    {
        int sum = 0;
        for(int i = 0; i < str.length(); i++)
        {
            sum = sum + (int)str.charAt(i);
        }
        return (float)sum/str.length();
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        String str = sc.next();
        System.out.printf("%.2f",average(str));
    }
}
```

Question-7

Problem Statement:

Level order traversal
Given a binary tree, find its level order traversal.
Level order traversal of a tree is breadth-first traversal for the tree
Your Task:
You don't have to take any input. Complete the function level Order() that takes the root node as input parameter and returns a list of integers containing the level order
traversal of the given Binary Tree.
Expected Time Complexity: O(N)
Expected Auxiliary Space: O(N)
Constraints:
1 ≤ Number of nodes ≤ 105
1 ≤ Data of a node ≤ 105

Question-7

Problem Statement:

```
Test Case #1
Input: 1
         / \
        3  2
Sample Output:
1 3 2

Test Case #2
Input: 10
          / \
        20 30
        / \
     40 60
Sample Output:
10 20 30 40 60
```

Solution

```
import java.util.LinkedList;
import java.util.Queue;
import java.io.*;
import java.util.*;
class Node{
int data;
Node left;
Node right;
Node(int data){
this.data = data;
left=null;
right=null;
} }
class LOT {
static Node buildTree(String str){
if(str.length()==0 || str.charAt(0)=='N'){
return null;
}
String ip[] = str.split(" ");
```

```
Node root = new
Node(Integer.parseInt(ip[0]));
Queue<Node> queue = new LinkedList<>();
queue.add(root);
int i = 1;
while(queue.size()>0 && i < ip.length) {
Node currNode = queue.peek();
queue.remove();
// Get the current node's value from the
string
String currVal = ip[i];
// If the left child is not null
if(!currVal.equals("N")) {
// Create the left child for the current
node
currNode.left = new
Node(Integer.parseInt(currVal));
// Push it to the queue
queue.add(currNode.left);
}
```

Solution

```
// For the right child
i++;
if(i >= ip.length)
break;
currVal = ip[i];
// If the right child is not null
if(!currVal.equals("N")) {
// Create the right child for the current
node
currNode.right = new
Node(Integer.parseInt(currVal));
// Push it to the queue
queue.add(currNode.right);
}
i++;
}
return root;
}
static void printInorder(Node root)
{
if(root == null)
return;
printInorder(root.left);
System.out.print(root.data+" ");
printInorder(root.right);
}
public static void main (String[] args)
throws IOException{
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
int t=Integer.parseInt(br.readLine());
while(t > 0){
String s = br.readLine();
Node root = buildTree(s);
Solution g = new Solution();
ArrayList <Integer> res =
g.levelOrder(root);
for (Integer num : res) System.out.print(num
+ " ");
System.out.println();
t--;
} } }
```

Solution

```
// } Driver Code Ends
//User function Template for Java
/*
class Node
{
int data;
Node left, right;
Node(int item)
{
data = item;
left = right = null;
} }
*/
class Solution
{
//Function to return the level order
traversal of a tree.
static ArrayList <Integer> levelOrder(Node
node)
{
ArrayList <Integer> a= new ArrayList
<Integer>();
Queue<Node> q= new LinkedList<Node>();
q.add(node);
while(!q.isEmpty()){
a.add(q.peek().data);
Node k=q.poll();
if(k.left!=null){
q.add(k.left);
}
if(k.right!=null){
q.add(k.right);
} }
return a;
} }
```

Question-8

Problem Statement:

Given an array of non-negative integers, and a value sum, determine if there is a subset of the given set with sum equal to given sum.

Your Task:
You don't need to read input or print anything. Your task is to complete the function
Is SubsetSum() which takes the array arr[], its size N and an integer sum as input parameters and returns boolean value true if there exists a subset with given sum and false otherwise.
The driver code itself prints 1, if returned value is true and prints 0 if returned value is
false.
Expected Time Complexity: O(sum*N)
Expected Auxiliary Space: O(sum*N)

Question-8

Problem Statement:

Test Case #1

Input:

N = 6

arr[] = {3, 34, 4, 12, 5, 2}

sum = 9

Sample Output: 1

Explanation: Here there exists a subset with sum = 9, 4+3+2 = 9.

Question-8

Problem Statement:

Test Case #2

Input:

N = 6

arr[] = {3, 34, 4, 12, 5, 2}

sum = 30

Sample Output: 0

Explanation: There is no subset with sum 30.

Solution

```
import java.io.*;
import java.util.*;
class SSP
{
public static void main(String args[])throws
IOException
{
BufferedReader read = new BufferedReader(new
InputStreamReader(System.in));
int t = Integer.parseInt(read.readLine());
while(t--> 0)
{
int N = Integer.parseInt(read.readLine());
String input_line[] =
read.readLine().trim().split("\\s+");
int arr[]= new int[N];
for(int i = 0; i < N; i++)
arr[i] = Integer.parseInt(input_line[i]);
int sum = Integer.parseInt(read.readLine());
Solution ob = new Solution();
if(ob.isSubsetSum(N, arr, sum))
System.out.println(1);
else
System.out.println(0);
}
}
}
// } Driver Code Ends
//User function Template for Java
class Solution{
static Boolean isSubsetSum(int N, int arr[],
int sum){ // code here
int dp=new int[N+1][sum+1];
for(int[] a:dp)
Arrays.fill(a,-1);
return util(arr,N-1,sum,dp);
}
```

Solution

```
static boolean util(int[] arr,int ind,int sum,int□□ dp)
{
if(ind==0)
return sum==arr[ind];
if(sum<0)
return false;
if(dp[ind][sum]!=-1)
return dp[ind][sum]==1;
if(sum==0)
return true;
boolean take= util(arr,ind-1,sum,dp);
boolean notTake=util(arr,ind-1,sum-arr[ind],dp);
if(take==true || notTake==true)
dp[ind][sum]=1;
else
dp[ind][sum]=0;
return take||notTake;
}}
```

Question-9

Problem Statement:

The stock span problem is a financial problem where we have a series of n daily price quotes for a stock and we need to calculate the span of stocks price for all n days.

The span Si of the stocks price on a given day i is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day.

For example, if an array of 7 days prices is given as {100, 80, 60, 70, 60, 75, 85}, then the span values for corresponding 7 days are {1, 1, 1, 2, 1, 4, 6}

```
prices = {100, 80, 60, 70, 60, 75, 85}
span = {1, 1, 1, 2, 1, 4, 6}.
```

Time Complexity: O(n)
Auxilliary Space Complexity: O(n)

Question-9

Problem Statement:

Test Case #1

Sample IO

Input

```
price[] = { 10, 4, 5, 90, 120, 80 };
```

Output

```
1 1 2 4 5 1
```

Solution

```
import java.util.Stack;
import java.util.Arrays;
public class EthCode {
// A stack based efficient method to
calculate // stock span values
static void calculateSpan(int price[], int
n, int S[]) {
// Create a stack and push index of first
element // to it
Stack <Integer> st = new Stack <>();
 st.push(0);
// Span value of first element is always 1
S[0] = 1;
// Calculate span values for rest of the
elements
for (int i = 1; i < n; i++) {
// Pop elements from stack while stack is
not
// empty and top of stack is smaller than
// price[i]
while (!st.empty() && price[st.peek()] <=
price[i])
st.pop();
// If stack becomes empty, then price[i] is
 // greater than all elements on left of it,
i.e.,
 // price[0], price[1], ..price[i-1]. Else
price[i]
// is greater than elements after top of
stack
S[i] = (st.empty()) ? (i + 1) : (i -
st.peek());
// Push this element to stack st.push(i);
static void printArray(int arr[]) {
System.out.print(Arrays.toString(arr));
}
public static void main(String[] args) {
int price[] = { 10, 4, 5, 90, 120, 80 };
int n = price.length;
int S[] = new int[n];
calculateSpan (price, n, S);
printArray(S);
```

Question-10

Problem Statement:

Compute the nearest larger number by interchanging its digits updated. Given 2 numbers a and b find the smallest number greater than b by interchanging the digits of a and if not possible print -1.

Question-10

Problem Statement:

Input Format

2 numbers a and b, separated by space.

Output Format

A single number greater than b.

If not possible, print -1

Constraints1 <= a,b <= 10000000

Question-10

Problem Statement:

Test Case #1
Sample 10-1
Input 459 500
Output 549
Test Case #2
Sample 10-2
Input
645757 457765
Output 465577

Solution

```
import java.util.*;
class Solution {
public static TreeSet<Integer> list = new
TreeSet <>(); static void smallestNumber
(String str, String ans)
{
if (str.length() == 0)
{
list.add (Integer.parseInt (ans));
return;
}
for (int i = 0; i < str.length (); i++)
char ch = str.charAt (i);
String ros = str.substring (0, i) +
str.substring (i +
1);
smallestNumber (ros, ans + ch);
}
}
```

```
public static void main(String[] args)
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
int b = sc.nextInt();
String s = a + "'";
smallestNumber (s, "");
Iterator itr = list.iterator(); int res = -
1;
while (itr.hasNext())
{
int no = (Integer) itr.next();
if (no > b)
{
res = no;
break;
}
}
System.out.println (res);
}
}
```

Question-11

Problem Statement:

Given a string str, a partitioning of the string is a palindrome partitioning if every sub-string of the partition is a palindrome. Determine the fewest cuts needed for palindrome partitioning of the given string.

Your Task:
You do not need to read input or print anything. Your task is to complete the function palindromicPartition() which takes the string str as the input parameter and returns the minimum number of partitions required.

Expected Time Complexity: O(n*n) [n is the length of the string str]
Expected Auxiliary Space: O(n*n)

Constraints:
1 ≤ length of str ≤ 500

Question-11

Problem Statement:

Test Case #1

Input: str = "ababbbabbababa"
Output: 3
Explanation: After 3 partitioning substrings are "a", "babbbab", "b", "ababa".

Test Case #2

Input: str = "aaabba"
Output: 1
Explanation: The substrings after 1 partitioning are "aa" and "abba".

Solution

```
import java.io.*;
import java.util.*;
class Place{
public static void main(String args[])throws
IOException
}
BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
int t = Integer.parseInt(in.readLine());
while(t-->0){
String str = in.readLine();
Solution ob = new Solution();
System.out.println(ob.palindromicPartition(s
tr));
}
}
}
// Driver Code Ends
//User function Template for Java
 public static void main(String[] args)
}
```

```
Scanner sc = new Scanner(System.in);
int a sc.nextInt();
int b = sc.nextInt();
String s = a + " ";
smallclass Solution{
static int palindromicPartition(String str)
{
int[][] dp = new
int[str.length()+1][str.length()+1];
for(int[] a: dp){
Arrays.fill(a,-1);
}
return solve(str, 0,str.length()-1, dp);
}
static int solve(String s, int i, int j,
int[][] dp}{
if(i>j)return 0;
if(isPalindrome(s,i,j))return 0;
if(dp[0]=-1)return dp[];
int res = Integer.MAX_VALUE;
```

Solution

```
for(int k = i; k<j; k++){
int temp = 1+ solve(s,i,k,dp) +
solve(s,k+1,j,dp);
if(temp<res)res = temp;
}
return dp[][]=res;
}
static boolean isPalindrome(String s, int i,
int){
if(i>j)return false;
if(i == j)return true;
 while(i<jX){
if(s.charAt(i)!=s.charAt(j)){
return false;
}
i++;
j--;
}
```

```
return true;
}
}
estNumber (s,"");
Iterator itr = list.iterator();
int res = -1; while (itr.hasNext())
}
int no =(Integer)itr.next();
if (no> b)
{
res = no; break;
System.out.println(res);
}
}
}
}
```