Ex. No. : Date :

Superman's Encounter

Problem Statement:

Superman is planning a journey to his home planet. It is very important for him to know which day he arrives there. They don't follow the 7-day week like us. Instead, they follow a 10-day week with the following days:

Day	Number Name of Day
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday
8	Kryptonday
9	Coluday
10	Daxamday

Here are the rules of the calendar:

- The calendar starts with Sunday always.
- It has only 296 days. After the 296th day, it goes back to Sunday.

You begin your journey on a Sunday and will reach after n. You have to tell on which day you will arrive when you reach there.

Input format:

- Contain a number n (0 < n)

Output format:

Print the name of the day you are arriving on

Sample Input

7

Sample Output

Kryptonday

Sample Input

1

Sample Output

Monday

Program:

```
#include<stdio.h>
int main()
{
    int n, day;
    scanf("%d", &n);
    if (n<296)
    day=n;
    else
    day=n-296;
    day%=10;
    day=day+1;
    day%=10;
    switch(day)
    {
    case 1: printf("Sunday"); break;
        case 2: printf("Monday"); break;
        case 3: printf("Tuesday"); break;
        case 4: printf("Wednesday"); break;
        case 5: printf("Thursday"); break;
        case 6: printf("friday"); break;
        case 7: printf("Saturday"); break;
        case 8: printf("Hrydonday"); break;
        case 9: printf("Coluday"); break;
        case 10: printf("Dascanday"); break;
    }
}
```

Result:

Thus the C program is implemented and executed successively.

Decision Making and Looping – while and do...while, for

Ex. No. :

Date :

Stone Game-One Four

Problem Statement:

Alice and Bob are playing a game called "Stone Game". Stone game is a two-player game. Let N be the total number of stones. In each turn, a player can remove either one stone or four stones. The player who picks the last stone, wins. They follow the "Ladies First" norm. Hence Alice is always the one to make the first move. Your task is to find out whether Alice can win, if both play the game optimally.

Input Format
First line starts with T, which is the number of test cases. Each test case will contain N number of stones.

Output Format
Print "Yes" in the case Alice wins, else print "No".

Constraints 1<=T<=1000 1<=N<=10000

Sample Input
3
1
6
7

Sample Output
Yes
Yes
No

Program:

```
#include<stdio.h>
int main()
{
    int T, i=0, n, t;
    scanf("%d", &T);
    while(i<T)
    {
        scanf("%d", &n);
       t=n/4 ;
        if(t%2==0 && n%2==0){
                printf("NO\n");
        }
        else if(t%2==1 && n%2==1)
        {
                printf("NO\n");
        }
        else
        {
                printf("YES\n");
        }
        i++;
    }
}
```

Ex. No. : **Date :**

Holes in a Number

Problem Statement:

You are designing a poster which prints out numbers with a unique style applied to each of them. The styling is based on the number of closed paths or holes present in a given number.

The number of holes that each of the digits from 0 to 9 have are equal to the number of closed paths in the digit. Their values are:

1, 2, 3, 5, 7	= 0 holes.
0, 4, 6, 9	= 1 hole.
8	= 2 holes.

Given a number, you must determine the sum of the number of holes for all of its digits. For example, the number 819 has 3 holes.
Complete the program, it must return an integer denoting the total number of holes in num.

Constraints
$1 \leq num \leq 109$

Input Format For Custom Testing
There is one line of text containing a single integer num, the value to process.

Sample Input
630

Sample Output
2

Program:

```
#include <stdio.h>
int main(){
    int a, b, n=0;
    scanf("%d", &a);
    while(a>0)
    {
        b = a%10;
        if (b==0 || b==6 || b==9 || b==4)
        {
            n=n+1;
        }
        else if(b==8){
            n=n+2;
        }
        a = a/10;
    }
    printf("%d", n);
}
```

Ex. No. :

Date :

Philaland Coin

Problem Statement:

The problem solvers have found a new Island for coding and named it as Philaland. These smart people were given a task to make a purchase of items at the Island easier by distributing various coins with different values. Manish has come up with a solution that if we make coins category starting from $1 till the maximum price of the item present on Island, then we can purchase any item easily. He added the following example to prove his point.

Let's suppose the maximum price of an item is 5$ then we can make coins of {$1, $2, $3, $4, $5}to purchase any item ranging from $1 till $5.

Now Manisha, being a keen observer suggested that we could actually minimize the number of coins required and gave following distribution {$1, $2, $3}. According to him any item can be purchased one time ranging from $1 to $5. Everyone was impressed with both of them. Your task is to help Manisha come up with a minimum number of denominations for any arbitrary max price in Philaland.

Input Format
Contains an integer N denoting the maximum price of the item present on Philaland.

Output Format
Print a single line denoting the minimum number of denominations of coins required.

Constraints
1<=T<=100 1<=N<=5000

Sample Input 1:
10

Sample Output 1:
4

Program:

```
#include <stdio.h>
int main () {
        int n,r = 0;
        scanf ("%d", &n);
        while (n != 0)
        {
            n = n/2;
            r = r+1;
        }
        printf ("%d", r);
}
```

Result:

Hence the C program is implemented and executed successfully

Ex. No. : **Date :**

Number Count

Problem Statement:

A set of N numbers (separated by one space) is passed as input to the program. The program must identify the count of numbers where the number is odd number.

Input Format:
The first line will contain the N numbers separated by one space.

Boundary Conditions:
3 <= N <= 50
The value of the numbers can be from -99999999 to 99999999

Output Format:
The count of numbers where the numbers are odd numbers.

Sample Input:
5 10 15 20 25 30 35 40 45 50

Sample Output:
5

Program:

```
#include<stdio.h>
int main(){
    int n, x=0;
    while(scanf("%d", &n)==1){
        if(n%2!=0){
            x++;}}
    printf("%d", x);
    return 0;
}
```

Ex. No. : **Date :**

Confusing Number

Problem Statement:

Given a number N, return true if and only if it is a *confusing number*, which satisfies the following condition:
We can rotate digits by 180 degrees to form new digits. When 0, 1, 6, 8, 9 are rotated 180 degrees, they become 0, 1, 9, 8, 6 respectively. When 2, 3, 4, 5 and 7 are rotated 180 degrees, they become invalid. A *confusing number* is a number that when rotated 180 degrees becomes a **different** number with each digit valid.

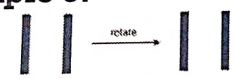
Example 1:

Input: 6
Output: true
Explanation: We get 9 after rotating 6, 9 is a valid number and 9 != 6.

Example 2:

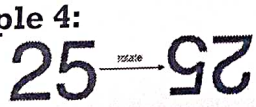
Input: 89
Output: true
Explanation: We get 68 after rotating 89, 86 is a valid number and 86 != 89.

Example 3:



Input: 11
Output: false
Explanation: We get 11 after rotating 11, 11 is a valid number but the value remains the same, thus 11 is not a confusing number.

Example 4:



Input: 25
Output: false
Explanation: We get an invalid number after rotating 25.

Note:

1. 0 <= N <= 10^9
2. After the rotation we can ignore leading zeros, for example if after rotation we have 0008 then this number is considered as just 8.

```
#include<stdio.h>
int main(){
    int n,x,y=1;
    scanf("%d",&n);
    while(n!=0 && y==1){
        x=n%10;n=n/10;
        if(x==2||x==3||x=4||x=7){
            y++;}}
    if(y==1){
        printf("true");}
    else{
        printf("false");}
}
```

Ex. No. :

Date :

Nutrition Value

Problem Statement:

A nutritionist is labeling all the best power foods in the market. Every food item arranged in a single line, will have a value beginning from 1 and increasing by 1 for each, until all items have a value associated with them. An item's value is the same as the number of macronutrients it has. For example, food item with value 1 has 1 macronutrient, food item with value 2 has 2 macronutrients, and incrementing in this fashion.

The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients. However, the nutritionist must avoid prescribing a particular sum of macronutrients (an 'unhealthy' number), and this sum is known. The nutritionist chooses food items in the increasing order of their value. Compute the highest total of macronutrients that can be prescribed to a patient, without the sum matching the given 'unhealthy' number.

Here's an illustration: Given *4* food items (hence value: *1,2,3* and *4*), and the unhealthy sum being *6* macronutrients, on choosing items *1, 2, 3* -> the sum is *6*, which matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is from among:

- *2 + 3 + 4 = 9*
- *1 + 3 + 4 = 8*
- *1 + 2 + 4 = 7*

Since *2 + 3 + 4 = 9*, allows for maximum number of macronutrients, 9 is the right answer. Complete the code in the editor below. It must return an integer that represents the maximum total of macronutrients, modulo *1000000007 (109 + 7)*.

It has the following:

n: an integer that denotes the number of food items

k: an integer that denotes the unhealthy number

Constraints

- $1 \leq n \leq 2 \times 109$
- $1 \leq k \leq 4 \times 1015$

Input Format For Custom Testing

The first line contains an integer, *n*, that denotes the number of food items. The second line contains an integer, *k*, that denotes the unhealthy number.

Sample Input 0

2
2

Sample Output 0

3

Program:

```
#include <stdio.h>
int main(){
    long long int n, t, i, sum=0;
    scanf("%lld %lld", &n, &t);
    for(i=1; i<=n; i++){
        sum=sum+i;
        if(sum==t){
            sum=sum-1;}}
    printf("%lld", sum%1000000007);
}
```

Result:

Hence the c program is implemented and executed sucessfully

Nested Loops - while and for, Jumps in Loops

Ex. No. :

Date :

Simple Chessboard

Problem Statement:

Write a program that prints a simple chessboard.

Input format:
The first line contains the number of inputs T.
The lines after that contain a different value for size of the chessboard

Output format:
Print a chessboard of dimensions size * size.
Print W for white spaces and B for black spaces.

Sample Input:
2
3
5

Sample Output:
WBW
BWB
WBW
WBWBW
BWBWB
WBWBW
BWBWB
WBWBW

Program:

```
#include<stdio.h>
int main(){
    int T, d, i=0, i1, i2, 0;
    char c;
    scanf("%d", &T);
    while(i1<&T)
    {
        scanf("%d", &d);
        i1=0;
        while(i1<d){
            0=1;
            i2=0;
            if(i1%2==0){ 0=0;}
            while(i2<d){
                c='B';
                if(i2%2==0){ c='W';}
                printf("%c", c);
                i2++;
            }
            i1+=1
            printf("\n");
        }
        i=i+1;
    }
}
```

Ex. No. :

Date :

Print Our Own Chessboard

Problem Statement:

Let's print a chessboard!

Write a program that takes input:
The first line contains T, the number of test cases
Each test case contains an integer N and also the starting character of the chessboard

Output Format
Print the chessboard as per the given examples

Sample Input:
2
2 W
3 B

Sample Output:
WB
BW
BWB
WBW
BWB

Program:

```
#include <stdio.h>
int main()
{
    int T,d,i,i1,i2,o,z;
    char c,s;
    scanf("%d",&T);
    for(i=0;i<T;i++)
    {
        scanf("%d %c",&d,&s);
        for(i1=0;i1<d;i1++)
        {
            z=(s=='W')?0:1;
            o=(i1%2==z)?0:1;
            for(i2=0;i2<d;i2++)
            {
                c=(i2%2==0)?'W':'B';
                printf("%c",c);
            }
            printf("\n");
        }
    }
    return 0;
}
```

Ex. No. :

Date :

Pattern Printing

Problem Statement:

Decode the logic and print the Pattern that corresponds to given input.

If N= 3 then pattern will be:
102030100111012
**4050809
****607
If N= 4, then pattern will be:
10203040170180190 20
**50607014015016
****809012013
******10011

Constraints: 2 <= N <= 100
Input Format
First line contains T, the number of test cases, each test case contains a single integer N

Output Format
First line print Case #i where i is the test case number, In the subsequent line, print the pattern

Sample Input
3
3
4
5

Sample Output
Case #1
102030100111012
**4050809
****607
Case #2
10203040170180190 20
**50607014015016
****809012013
******10011
Case #3
1020304050260270280290 30
**6070809022023024025
****10011012019020021
******13014017018
********15016

Program:

```
#include <stdio.h>
int main(){
        int n,v,p3,c,in,i,i1,i2,t,ti;
        scanf("%d",&t);
        for(ti=0; ti<t; ti++){
            v=0;
            scanf("%d",&n);
            printf("Case #%d\n", ti+1);
            for(i=0;i<n;i++){
                c=0;
                if(i>0){ for(i1=0;i1<i;i++)printf("**");}
            for(i1=i;i1<n;i1++){
                if(i>0)c++;
                printf("%d 0^",++v);
            }
            if(i==0){ p3=v+(v*(v-1))+1; in=p3; }
            in = in - c;
            p3 = in;
            for(i2=i; i2<n;i++){
                printf("%d", p3++);
                if(i2!=n-1) printf("0");
            } printf("\n");
            }
    } }
```

Result: Hence the C program is implemented and executed successfully

Ex. No. :

Date :

Armstrong Number

Problem Statement:

The k-digit number N is an Armstrong number if and only if the k-th power of each digit sums to N.

Given a positive integer N, return true if and only if it is an Armstrong number.

Note: 1 <= N <= 10^8

Hint: 153 is a 3-digit number, and 153 = 1^3 + 5^3 + 3^3.

Sample Input:
153

Sample Output:
true

Sample **Input:**
123

Sample **Output:**
false

Sample **Input:**
1634

Sample **Output:**
true

Program:

```
#include <stdio.h>
int main (){
    int n;
    scanf ("%d", &n );
    int x=0, n2= n;
    while(n2 != 0)
    {
        x++;
        n2 = n2/10;
    }
    int sum = 0;
    int n3=n, n4;
    while (n3!=0)
    {
        n4 = n3 % 10;
        sum = sum + pow (n4, x);
        n3 = n3/10;
    }
    if (n == sum)
    {
        printf ("true");
    }
    else
    {
        printf ("false");
    }
    return 0;
}
```

Ex. No. : **Date :**

Reverse and Add Until Get a Palindrome

Problem Statement:

Take a number, reverse it and add it to the original number until the obtained number is a palindrome.

Constraints
1<=num<=99999999

Sample Input 1
32

Sample Output 1
55

Sample Input 2
789

Sample Output 2
66066

Program:

```
#include <stdio.h>
int main ()
{
    int rn, n, nt = 0, i = 0,
    scanf ("%od ", &n);
    do {
        nt = n; rn = 0;
        while(n != 0)
        {
            rn = rn * 10 + n % 10;
            n = n / 10;
        }
        n = nt + rn;
        i++;
    }
    while (rn != nt || i == 1);
    printf("%od", rn);
    return 0;
}
```

Ex. No. :

Date :

Lucky Number

Problem Statement:

A number is considered lucky if it contains either 3 or 4 or 3 and 4 both in it. Write a program to print the nth lucky number. Example, 1st lucky number is 3, and 2nd lucky number is 4 and 3rd lucky number is 33 and 4th lucky number is 34 and so on. Note that 13, 40 etc., are not lucky as they have other numbers in it.

The program should accept a number 'n' as input and display the nth lucky number as output.

Sample Input 1:
3

Sample Output 1:
33

Program:

```
#include<stdio.h>
int main(){
    int n=1,i=0,nt,co=0,e;
    scanf("%d"&e);
    while(i<e)
    {
        nt=n;
        while(nt!=0)
        {
            co=0;
            if(nt%10!=3&&nt%10!=4).
            {
                co=1;
                break;
            }
            nt=nt/10;
        }
        if(co==0)
        {
            i++;
        }
        n++;
}
```

```
        printf ("%d", --n);
        return 0;
    }
```

Result:

The program is implemented and executed successfully.

One-Dimensional Arrays

Ex. No. : **Date :**

Check pair with difference k

Problem Statement:

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[i] - A[j] = k, i != j.

Input Format
1. First line is number of test cases T. Following T lines contain:
2. N, followed by N integers of the array
3. The non-negative integer k

Output format
Print 1 if such a pair exists and 0 if it doesn't.

Sample Input:
1
3 1 3 5
4

Sample Output:
1

Program:

```
#include<stdio.h>
int main(){
    int t;
    scanf("%d",&t);
    while(t--){
        int n;
        scanf("%d",&n);
        int a[n];
        for(int i=0;i<n;i++){
            scanf("%d",&a[i]);
        }
        int k;
        scanf("%d",&k);
        int flag=0;
        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                if(a[i]-a[j]==k||a[j]-a[i]==k)
                {flag=1;break;}
            }
        if(flag)break;}
        printf("%d\n",flag);
    }
}
```

Ex. No. : **Date :**

Chocolates

Problem Statement:

Sam loves chocolates and starts buying them on the 1st day of the year. Each day of the year, x, is numbered from 1 to Y. On days when x is odd, Sam will buy x chocolates; on days when x is even, Sam will not purchase any chocolates.

Complete the code in the editor so that for each day Ni (where $1 \leq x \leq N \leq Y$) in array arr, the number of chocolates Sam purchased (during days 1 through N) is printed on a new line. This is a function-only challenge, so input is handled for you by the locked stub code in the editor.

Input Format
The program takes an array of integers as a parameter.

The locked code in the editor handles reading the following input from stdin, assembling it into an array of integers (arr), and calling calculate(arr).

The first line of input contains an integer, T (the number of test cases). Each line i of the T subsequent lines describes the ith test case as an integer, Ni (the number of days).

Constraints
$1 \leq T \leq 2 \times 105$
$1 \leq N \leq 2 \times 106$
$1 \leq x \leq N \leq Y$

Output Format
For each test case, Ti in arr, your calculate method should print the total number of chocolates Sam purchased by day Ni on a new line.

Sample Input 0
3
1
2
3

Sample Output 0
1
1
4

Program:

```
#include <stdio.h>
int main(){
    int t;
    scanf("%d", &t);
    while(t--){
        int n, c=0;
        scanf("%d", &n);
        for(int i=0 ; i<=n; i++){
            if(i%2!=0) c=c+i;
        } printf("%d\n", c);
    }
}
```

Ex. No. : Date :

Football Scores

Problem Statement:

The number of goals achieved by two football teams in matches in a league is given in the form of two lists. Consider:

- Football team A, has played three matches, and has scored { 1 , 2 , 3 } goals in each match respectively.
- Football team B, has played two matches, and has scored { 2, 4 } goals in each match respectively.
- Your task is to compute, for each match of team B, the total number of matches of team A,
- where team A has scored less than or equal to the number of goals scored by team B in that match.

In the above case:

- For 2 goals scored by team B in its first match, team A has 2 matches with scores 1 and 2.
- For 4 goals scored by team B in its second match, team A has 3 matches with scores 1, 2 and 3. Hence, the answer: {2, 3}.

Complete the code in the editor below. The program must return an array of m positive integers, one for each maxes[i] representing the total number of elements nums[j] satisfying nums[j] ≤ maxes[i] where $0 \le j < n$ and $0 \le i < m$, in the given order.

It has the following:
nums[nums[0],...nums[n-1]]: first array of positive integers
maxes[maxes[0],...maxes[n-1]]: second array of positive integers

Constraints:
$2 \le n, m \le 105$, $1 \le nums[j] \le 109$, where $0 \le j < n$, $1 \le maxes[i] \le 109$, where $0 \le i < m$.

Input Format For Custom Testing
Input from stdin will be processed as follows and passed to the function.
The first line contains an integer n, the number of elements in nums.
The next n lines each contain an integer describing nums[j] where $0 \le j < n$.
The next line contains an integer m, the number of elements in maxes.
The next m lines each contain an integer describing maxes[i] where $0 \le i < m$.

Sample Input
4
1
4
2
4
2
3
5

Sample Output
2
4

Program:

```
#include <stdio.h>
int main()
{
    int s1, s2, ans;
    scanf ("%d", &s1);
    int ta [s1];
    for (int i=0, i<s1; i++)
    scanf ("%d", &s2);
    int tb [s2];
    for(int i=0; i<s2; i++)
    scanf ("%d", &s2);
    int tb[s2];
    for(int j=0; j<s2; j++)
    {
        ans=0;
        for (int i=0; i<s1; i++){
            if (tb [j] >= ta [i])
            ans++;
        } printf ("%d\n", ans);
    }
}
```

Searching Algorithms – Linear and Binary

Ex. No. :

Date :

Ice Cream Parlor

Problem Statement:

Sunny and Johnny like to pool their money and go to the ice cream parlor. Johnny never buys the same flavor that Sunny does. The only other rule they have is that they spend all of their money.

Given a list of prices for the flavors of ice cream, select the two that will cost all of the money they have.

For example, they have m = 6 to spend and there are flavors costing cost = [1, 2, 3, 4, 5, 6]. The two flavors costing 1 and 5 meet the criteria. Using 1-based indexing, they are at indices 1 and 4.

Complete the code in the editor below. It should return an array containing the indices of the prices of the two flavors they buy, sorted ascending.

It has the following:

- m: an integer denoting the amount of money they have to spend
- cost: an integer array denoting the cost of each flavor of ice cream

Input Format

The first line contains an integer, t, denoting the number of trips to the ice cream parlor. The next t sets of lines each describe a visit. Each trip is described as follows:

1. The integer m, the amount of money they have pooled.
2. The integer n, the number of flavors offered at the time.
3. n space-separated integers denoting the cost of each flavor: cost[cost[1], cost[2], . . . , cost[n]].

Note: The index within the cost array represents the flavor of the ice cream purchased.

Constraints

- $1 \leq t \leq 50$
- $2 \leq m \leq 104$
- $2 \leq n \leq 104$
- $1 \leq cost[i] \leq 104, \forall i \in [1, n]$
- There will always be a unique solution.

Output Format

For each test case, print two space-separated integers denoting the indices of the two flavors purchased, in ascending order.

Sample Input

```
2
4
5
1 4 5 3 2
4
4
2 2 4 3
```

Sample Output 1 4

```
1 2
```

Program:

```
#include<stdio.h>
int main(){
    int t,m,n,c=0,
    scanf("%d",&t);
    for(int i=0;i<t;i++){
        c=0;
        scanf("%d\n%d",&m,&n);
        int arr[n];
        for(int j=0;j<n;j++){
            scanf("%d",&arr[j]);
        }
        for(int a=0;a<n-1;a++){
            for(int b=a+1;b<n;b++){
                if(arr[a]+arr[b]==m){
                    printf("%d %d\n",a+1,b+1);
                    c=1;break;
                }
            } if(c==1)break;
        }
    }
return 0;}
```

Ex. No. : **Date :**

Missing Numbers

Problem Statement:

Numeros the Artist had two lists that were permutations of one another. He was very proud. Unfortunately, while transporting them from one exhibition to another, some numbers were lost out of the first list. Can you find the missing numbers?

As an example, the array with some numbers missing, arr = [7, 2, 5, 3, 5, 3]. The original array of numbers brr = [7, 2, 5, 4, 6, 3, 5, 3]. The numbers missing are [4, 6].

Notes

- If a number occurs multiple times in the lists, you must ensure that the frequency of that number in both lists is the same. If that is not the case, then it is also a missing number.
- You have to print all the missing numbers in ascending order.
- Print each missing number once, even if it is missing multiple times.
- The difference between maximum and minimum number in the second list is less than or equal to 100.

Complete the code in the editor below. It should return a sorted array of missing numbers. It has the following:

- arr: the array with missing numbers
- brr: the original array of numbers

Input Format

There will be four lines of input:
n - the size of the first list, arr
The next line contains n space-separated integers arr[i]
m - the size of the second list, brr
The next line contains m space-separated integers brr[i]

Constraints

$1 \leq n, m \leq 2 \times 105, n \leq m, 1 \leq brr[i] \leq 2 \times 104, Xmax - Xmin < 101$

Output Format

Output the missing numbers in ascending order.

Sample Input

```
10
203 204 205 206 207 208 203 204 205 206
13
203 204 204 205 206 207 205 208 203 206 205 206 204
```

Sample Output

```
204 205 206
```

Program:

```
#include<stdio.h>
int main(){
    int n,m,c,c1=0,c0;
    scanf("%d",&n);
    int arr[n];
    for(int a=0;a<n;a++){
        scanf("%d",&arr[a]);
    }
    scanf("%d",&n);
    int brr[m],ans[m];
    for(int b=0;b<m;b++){
        scanf("%d",&brr[b]);
    }
    for(int j=0;j<m;j++)
    {
        c=0;
        for(int i=0;i<n;i++){
            if(arr[i]==brr[j]){
                c=1;
                arr[i]=-1;
                break;
            }
        }
        if(c==0){
            ans[c1]=brr[j];
```

```
            c1++;
        }
    }
    for (int a=0; a<c1; a++){
        c0 = 0;
        for (int b=0; b<c1; b++){
            if (ans[b] < ans[c0])
                c0++;
        }
        int temp = ans[a];
        ans[a] = ans[c0];
        ans[c0] = temp;
    }
    for (int i=0; i<c1; i++)
    printf ("%d", ans[i]);
    return 0;
}
```

Ex. No. : **Date :**

Sherlock and Array

Problem Statement:

Watson gives Sherlock an array of integers. His challenge is to find an element of the array such that the sum of all elements to the left is equal to the sum of all elements to the right. For instance, given the array arr = [5, 6, 8, 11], 8 is between two subarrays that sum to 11. If your starting array is [1], that element satisfies the rule as left and right sum to 0. You will be given arrays of integers and must determine whether there is an element that meets the criterion.

Complete the code in the editor below. It should return a string, either YES if there is an element meeting the criterion or NO otherwise. It has the following: arr: an array of integers

Input Format
The first line contains T, the number of test cases.
The next T pairs of lines each represent a test case.
- The first line contains n, the number of elements in the array arr.
- The second line contains n space-separated integers arr[i] where $0 \leq i < n$.

Constraints: $1 \leq T \leq 10$, $1 \leq n \leq 105$, $1 \leq arr[i] \leq 2 \times 104$, $0 \leq i \leq n$

Output Format
For each test case print YES if there exists an element in the array, such that the sum of the elements on its left is equal to the sum of the elements on its right; otherwise print NO.

Sample Input 0
2
3
1 2 3
4
1 2 3 3

Sample Output 0
NO
YES

Program:

```
#include<stdio.h>
int main(){
    int t,n,Is, rs, m;
    scanf("%d",&t);
    for(int i=0; i<t; i++){
        Is =0;
        rs =0;
        scanf("%d",&n);
        int arr[n];
        for(int j=0 ; j<n ; j++)
            scanf("%d", &arr [j]);

    m =n/2 ;
    if(arr[m]==0){
            for(m=0; arr [m]==0 && m<n ;m++);
    }
    for(int j=0 ; j<=m; j++)
    Is =Is +arr[j]
    for(int j =m; j<n ; j++)
    rs= rs+ arr[j] ;
    printf("%s\n", (Is ==rs)?"YES":NO");
    }
    return 0;
}
```

Result:

Thus the C program is executed and implemented successfully.

Sorting Algorithms – Bubble and Selection

Ex. No. :

Date :

Easy Going

Problem Statement:

Coders here is a simple task for you, you have given an array of size N and an integer M. Your task is to calculate the difference between maximum sum and minimum sum of N-M elements of the given array.

Constraints:
1<=t<=10
1<=n<=1000
1<=a[i]<=1000

Input Format:
First line contains an integer T denoting the number of testcases.
First line of every testcase contains two integer N and M.
Next line contains N space separated integers denoting the elements of array

Output:
For every test case print your answer in new line

Sample Input
1
5 1
1 2 3 4 5

Sample Output
4

Explanation

M is 1 and N is 5 so you have to calculate maximum and minimum sum using (5-1 =) 4 elements.
Maximum sum using the 4 elements would be (2+3+4+5=)14.
Minimum sum using the 4 elements would be (1+2+3+4=)10.
Difference will be 14-10=4.

Program:

```
#include <stdio.h>
int main(){
        int t;
        scanf("%d", &t);
        while(t--)
        {
            int n,m,d, min, temp;
            scanf("%d %d", &n, &m);
            d=n - m;
            int arr[n];
            for(int i=0; i<n; i++)
            scanf("%d", &arr[i]);
            for(int j=0; j<n; j++)
            {
                min=j
                for(int k = j; k<n; k++)
                {
                    if(arr[k]<arr[min])
                    min=k;
                }
                temp = arr[min];
                arr[min]=arr[j];
                arr[j] = temp;
            }
            int maxsum=0, minsum =0;
            for(int a = 0; a< d; a++)
            minsum +=arr[a];
```

```
for (int b=n-1; b>=m-1; b--)
max = cur[b];
printf("%d\n", maximum - min sum);
    }
}
```

Ex. No. :

Date :

Sort it out!

Problem Statement:

You are given an array A of non-negative integers of size m. Your task is to sort the arra
in nondecreasing order and print out the original indices of the new sorted array.

Example:
A={4,5,3,7,1}
After sorting the new array becomes A={1,3,4,5,7}.
The required output should be "4 2 0 1 3"

Input Format:
The first line of input consists of the size of the array
The next line consists of the array of size m

Output Format:
Output consists of a single line of integers

Constraints:
1<=m<=106
0<=A[i]<=106
NOTE: The indexing of the array starts with 0.

Sample Input
5
4 5 3 7 1

Sample Output
4 2 0 1 3

Program:

```
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++)
    scanf("%d",&arr[i]);
    int max=arr[0];
    for(int i=1; i<n;i++)
    {
        if(arr[i]>max)
        max=arr[i];
    }
    max++;
    int min=0;
    for(int a=0; a<n;a++)
    {
        for(int b=0; b<n; b++)
        {
            if(arr[b]<arr[min])
            min=b;
        }
        printf("%d", min);
        arr[min]=max;
    }
}
```

Ex. No. : **Date :**

Save Patients

Problem Statement:

A new deadly virus has infected large population of a planet. A brilliant scientist has discovered a new strain of virus which can cure this disease. Vaccine produced from this virus has various strength depending on midichlorians count. A person is cured only if midichlorians count in vaccine batch is more than midichlorians count of person. A doctor receives a new set of report which contains midichlorians count of each infected patient, Practo stores all vaccine doctor has and their midichlorians count. You need to determine if doctor can save all patients with the vaccines he has. The number of vaccines and patients are equal.

Input Format
First line contains the number of vaccines - N. Second line contains N integers, which are strength of vaccines. Third line contains N integers, which are midichlorians count of patients.

Output Format
Print a single line containing 'Yes' or 'No'.

Input Constraint
$1 < N < 10$
Strength of vaccines and midichlorians count of patients fit in integer.

Sample Input
5
123 146 454 542 456
100 328 248 689 200

Sample Output
No

Program:

```
#include <stdio.h>
int main()
{
    int n,min1, min2,temp, flag=1;
    scanf("%d", &n);
    int nac[n], pat[n];
    for (int i=0; i <n;i++)
    scanf("%d", &nac[i]);
    for(int i =0; i<n;i++)
    scanf("%d", &pat[i]);
    for(int j=0; j<n-1; j++)
    {
        min1 = j, min2 = j;
        for (int k=j ; k<n; k++)
        {
            if (nac[k] <nac[min1])
            min1 = k;
            if (pat[k]<pat[min2])
            min2=k;
        }
    temp =nac[min1];
    nac[min1] = nac[j];
    nac[j] =temp;
    temp = pat[min2];
    pat[min2] = pat[j];
```

```
        pat[j] = temp;
    }
    for (int i=0; i<n; i++)
    {
        if (arr[i] <= pat[i])
        {
            flag=0;
            break;
        }
    }
    if (flag==1)
    printf("Yes");
    else
    printf("No");
}
```

Ex. No. :

Date :

Shubham and Xor

Problem Statement:

You are given an array of n integer numbers a1, a2, . . . , an. Calculate the nu
of indices (i, j) such that $1 \leq i < j \leq n$ and ai xor aj = 0.

Input format
- First line: n denoting the number of array elements
- Second line: n space separated integers a1, a2, . . . , an.

Output format
Output the required number of pairs.

Constraints
$1 \leq n \leq 106$
$1 \leq ai \leq 109$

Sample Input
5
1 3 1 4 3

Sample Output
2

Explanation
The 2 pair of indices are (1, 3) and (2,5).

Program:

```
#include<stdio.h>
int main()
{
    int n, count=0;
    scanf("%d", &n);
    int arr[n];
    for(int i=0; i<n; i++)
    scanf("%d", &arr[i]);
    for(int i=0; i<n-1; i++)
    {
        for(int j=i+1; j<n; j++)
        {
            if((arr[i]^arr[j]==0)
            count++;
        }
    }
    printf("%d", count);
}
```

Result :

Thus, the program is completely implemented and executed successfully