**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("%d", &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("%d", 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;
}
```

**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+1;
        }
        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<s; i++) {
        scanf("%d", &ta[i]);
        scanf("%d", &s2);
        int tb[s2]; }
    for (int i = 0; i<s2 ; i++) {
    scanf("%d", &tb[i]);
    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);
    }
}
```

**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", &m);
    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[a])
            co++;
        }
        int temp = ans[a];
        ans[a] = ans[co];
        ans[co] = 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, ls, rs, m;
    scanf("%d", &t);
    for(int i=0; i<t; i++){
        ls=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++)
    ls = ls + arr[j];
    for(int j=m; j<n; j++)
    rs = rs + arr[j];
    printf("%s\n", (ls==rs)?"YES":"NO");
    }
  return 0;
}
```

**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 > n-1; b--)
maxsum += arr[b];
printf("%d\n", maxsum - minsum);
```

**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 array 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;
    }
}
```