

Structure Pointers & Array Pointers LEVEL 1 KV

```
#include<stdio.h>
#include<string.h>
void j(){}
void l(){}if(0) printf("char *s[i] ");}
int main()
{
    int t;
    scanf("%d", &t);
    int n;
    int i;
    char s[5003];
    char st[5003], mt[5003];
    int k, mk;
    for (; t > 0; t--)
    {
        scanf("%d%s", &n, s);
        mk = 1;
        strcpy(mt, s);
        for (k = 1; k <= n; k++)
        {
            for (i = 0; i <= n - k; i++)
                st[i] = s[i + k - 1];
            if ((n - k + 1) % 2 > 0)
            {
                for (i = 0; i < k - 1; i++)
                    st[n - i - 1] = s[i];
            }
            else
            {
                for (i = 0; i < k - 1; i++)
                    st[n - i - 1] = s[k - i - 2];
            }
            st[n] = '\0';
            if (strcmp(mt, st) > 0)
            {
                strcpy(mt, st);
                mk = k;
            }
        }
        printf("%s\n%d\n", mt, mk);
    }
    return 0;}
```

Untitled document - Google Docs x (2) WhatsApp x srmktretelab x

care.srmist.edu.in/srmktretelab/#/srmktretelab/student/home

Apps Gmail YouTube Maps codetantra GCR Google Meet student login Academia CIVIL AND MECH srmktretelab C compiler empower WhatsApp Reading list

CHALLENGE INFORMATION

You have already solved this challenge ! Though you can run the code with different logic !

Course	C	Session	Structure Pointers & Array Pointers	Question Information	Level 1 Challenge 72
		<p>Problem Description</p> <p>Priya got a new doll these days. It can even walk! Priya has built a maze for the doll and wants to test it. The maze is a grid with n rows and m columns. There are k obstacles, the i-th of them is on the cell (x_i, y_i), which means the cell in the intersection of the x_i-th row and the y_i-th column.</p> <p>However, the doll is clumsy in some ways. It can only walk straight or turn right at most once in the same cell (including the start cell). It cannot get into a cell with an obstacle or get out of the maze.</p> <p>More formally, there exist 4 directions, in which the doll can look:</p> <ol style="list-style-type: none"> The doll looks in the direction along the row from the first cell to the last. While moving looking in this direction the doll will move from the cell (x, y) into the cell $(x, y+1)$; The doll looks in the direction along the column from the first cell to the last. While moving looking in this direction the doll will move from the cell (x, y) into the cell $(x+1, y)$; The doll looks in the direction along the row from the last cell to first. While moving looking in this direction the doll will move from the cell (x, y) into the cell $(x, y-1)$; The doll looks in the direction along the column from the last cell to the first. While moving looking in this direction the doll will move from the cell (x, y) into the cell $(x-1, y)$. <p>Standing in some cell the doll can move into the cell in the direction it looks or it can turn right once. Turning right once, the doll switches it's direction by the following rules: $1 \rightarrow 2$, $2 \rightarrow 3$, $3 \rightarrow 4$, $4 \rightarrow 1$. Standing in one cell, the doll can make at most one turn right.</p> <p>Now Priya is controlling the doll's moves. She puts the doll in of the cell $(1, 1)$ (the upper-left cell of the maze). Initially, the doll looks to direction 1, so along the row from the first cell to the last. She wants to let the doll walk across all the cells without obstacles exactly once and end in any place. Can it be achieved?</p> <p>Constraints</p> <p>$1 \leq n, m \leq 10^4$, $0 \leq k \leq 10^4$ $1 \leq x_i \leq n$, $1 \leq y_i \leq m$</p> <p>Input Format</p>			

```
#include <stdlib.h>
```

```
#define N 100000
#define M 100000
#define K 100000
```

```
int min(int a, int b) { return a < b ? a : b; }
int max(int a, int b) { return a > b ? a : b; }
```

```
int move(int *aa, int k, int j0, int j1, int incr) {
    int j_, h;

    j_ = -1;
    for (h = 0; h < k; h++) {
        int j = aa[h];

        if (j < j0 || j > j1)
            continue;
        j_ = j_ == -1 ? j : incr ? min(j_, j) : max(j_, j);
    }
    return j_ == -1 ? j1 - j0 + 1 : incr ? j_ - j0 : j1 - j_;
}
```

```
int main() {
    static int *aa[N], ka[N], *bb[N], kb[M], ii[K], jj[K];
    int n, m, k, h, i, j, i0, i1, j0, j1, d_;
    long long sum;

    scanf("%d%d%d", &n, &m, &k);
    for (h = 0; h < k; h++) {
        scanf("%d%d", &i, &j), i--, j--;
        ii[h] = i, jj[h] = j;
        ka[i]++, kb[j]++;
    }
    for (i = 0; i < n; i++) {
        aa[i] = malloc(ka[i] * sizeof *aa[i]);
        ka[i] = 0;
    }
    for (j = 0; j < m; j++) {
        bb[j] = malloc(kb[j] * sizeof *bb[j]);
        kb[j] = 0;
    }
    for (h = 0; h < k; h++) {
        i = ii[h], j = jj[h];
        aa[i][ka[i]++] = j;
        bb[j][kb[j]++] = i;
    }
    i0 = 0, i1 = n - 1, j0 = 0, j1 = m - 1, d_ = 1;
    sum = 0;
    while (i0 <= i1 && j0 <= j1) {
        int cnt;

        if (d_ == 1) {
            if ((cnt = move(aa[i0], ka[i0], j0, j1, 1)) == 0)
                break;

            i0++;
            j1 = j0 + cnt - 1;
        } else if (d_ == 2) {
            if ((cnt = move(bb[j1], kb[j1], i0, i1, 1)) == 0)
                break;

            j1--;
            i1 = i0 + cnt - 1;
        } else if (d_ == 3) {

```

```

        if ((cnt = move(aa[i1], ka[j1], j0, j1, 0)) == 0)
            break;

        i1--;
        j0 = j1 - cnt + 1;

    } else {

        if ((cnt = move(bb[j0], kb[j0], i0, i1, 0)) == 0)
            break;

        j0++;
        i0 = i1 - cnt + 1;

    }

    sum += cnt;
    if (d_++ == 4)
        d_ = 1;

}

printf(sum + k == (long long) n * m ? "Yes\n" : "No\n");
return 0;
}

```

The screenshot shows a web browser window with the URL `care.srmstedu.in/srmkretelab/#/srmkretelab/student/home`. The browser's address bar and tabs are visible. The page content includes a student profile with the following details:

- role: student
- name: kushagra
- ID: 782946222059
- dept: school of computing
- July 1st 2021, 8:12:47 pm
- Logout button

Below the profile, there is a "CHALLENGE INFORMATION" section. It contains a message: "You have already solved this challenge! Though you can run the code with different logic!". Below this, there is a table with the following columns: Course, Session, Structure Pointers & Array Pointers, Question Information, and Level 1 Challenge 73. The table contains the following information:

Course	Session	Structure Pointers & Array Pointers	Question Information	Level 1 Challenge 73
Problem	Problem Description: Dr. Abdul Kalam is a Professor at a top university. There are n students under Kalam supervision, the programming skill of the i -th student is a_i .			
	Kalam has to form k teams for yet another new programming competition. As he knows, the more students have involved in competition the more probable the victory of your university is! So Kalam has to form no more than k (and at least one) non-empty team so that the total number of students in them is maximized. But Kalam also knows that each team should be balanced. It means that the programming skill of each pair of students in each team should differ by no more than 5. Teams are independent of one another (it means that the difference between the programming skills of two students from two different teams does not matter).			
	It is possible that some students not be included in any team at all. Your task is to report the maximum possible total number of students in no more than k (and at least one) non-empty balanced teams.			
	Constraints: $1 \leq k \leq n \leq 5000$ $1 \leq a_i \leq 10^9$			
	Input Format: The first line of the input contains two integers n and k — the number of students and the maximum number of teams, correspondingly. The second line of the input contains n integers a_1, a_2, \dots, a_n , where a_i is a programming skill of the i -th student.			

```

#include <stdio.h>
#include <stdlib.h>

#define N 5000

int max(int a, int b) { return a > b ? a : b; }

int compare(const void *a, const void *b) {
    int ia = *(int *) a;
    int ib = *(int *) b;

    return ia - ib;
}

int main() {
    static int aa[N], dp[N + 1][N + 1];
    int n, k, h, i, j;

    scanf("%d%d", &n, &k);
    for (i = 0; i < n; i++)
        scanf("%d", &aa[i]);
    qsort(aa, n, sizeof *aa, compare);
    for (i = 0, j = 1; j <= n; j++) {
        while (aa[i] + 5 < aa[j] - 1)
            i++;
        for (h = 1; h <= k; h++)
            dp[j][h] = max(dp[j - 1][h], dp[i][h - 1] + j - i);
    }
    printf("%d\n", dp[n][k]);
    return 0;
}

```

Untitled document - Google Doc x (3) WhatsApp x srmktretelab x

care.srmist.edu.in/srmktretelab/#/srmktretelab/student/home

Apps Gmail YouTube Maps codetantra GCR Google Meet student login Academia CIVIL AND MECH srmktretelab C compiler empower WhatsApp Reading list

CHALLENGE INFORMATION

You have already solved this challenge ! Though you can run the code with different logic !

Course	C	Session	Structure Pointers & Array Pointers	Question Information	Level 1 • Challenge 74
			<p>Problem Description</p> <p>Recently, Bharani met with Dharani in a philatelic store, and since then they are collecting coins together. Their favorite occupation is to sort collections of coins. Dharani likes having things in order, that is why he wants his coins to be arranged in a row in such a way that firstly come coins out of circulation, and then come coins still in circulation.</p> <p>For arranging coins Bharani uses the following algorithm. One step of his algorithm looks like the following:</p> <p>He looks through all the coins from left to right; If he sees that the i-th coin is still in circulation, and $(i + 1)$-th coin is already out of circulation, he exchanges these two coins and continues watching coins from $(i + 1)$-th.</p> <p>Bharani repeats the procedure above until it happens that no two coins were exchanged during this procedure. Bharani calls hardness of ordering the number of steps required for him according to the algorithm above to sort the sequence, e.g. the number of times he looks through the coins from the very beginning. For example, for the ordered sequence hardness of ordering equals one.</p> <p>Today Dharani invited Bharani and proposed him a game. First he puts n coins in a row, all of them are out of circulation. Then Dharani chooses one of the coins out of circulation and replaces it with a coin in circulation for n times. During this process Dharani constantly asks Bharani what is the hardness of ordering of the sequence.</p> <p>The task is more complicated because Bharani should not touch the coins and he should determine hardness of ordering in his mind. Help Bharani with this task.</p> <p>Constraints</p> <p>$1 \leq n \leq 300\,000$ $1 \leq p_i \leq n$</p> <p>Input Format</p> <p>The first line contains single integer n — number of coins that Dharani puts behind Bharani.</p> <p>Second line contains n distinct integers p_1, p_2, \dots, p_n — positions that Dharani puts coins in circulation to. At first Dharani replaces coin located at position p_1, then coin located at position p_2 and so on. Coins are numbered from left to right.</p>		

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n,*q,x;
    scanf("%d",&n);
    q=(int*)calloc(n+1, sizeof(int));
    printf("1 ");
    int p=n,i;
    for( i=1;i<=n;i++)
    {
        scanf("%d",&x);
        q[x]=1;
        while(q[p]==1)
            p--;
        printf("%d ",i-n+p+1);return 0;}
}
```

(3) WhatsApp x srmktretelab x

care.srmist.edu.in/srmktretelab/#/srmktretelab/student/home

Apps Gmail YouTube Maps codetantra GCR Google Meet student login Academia CIVIL AND MECH srmktretelab C compiler empower WhatsApp Reading list

SRM role student name kushagra ID 782946222059 dept school of computing July 1st 2021, 8:13:50 pm Logout

CHALLENGE INFORMATION

You have already solved this challenge ! Though you can run the code with different logic !

Course	C	Session	Structure Pointers & Array Pointers	Question Information	Level 1 • Challenge 75
			<p>Problem Description</p> <p>Ramesh have been given an array A of size N and an integer K. This array consists of N integers ranging from 1 to 10^4. Each element in this array is said to have a Special Weight. The special weight of an element $a[i]$ is $a[i]^K$.</p> <p>Ramesh now need to sort this array in Non-Increasing order of the weight of each element, i.e the element with the highest weight should appear first, then the element with the second highest weight and so on. In case two elements have the same weight, the one with the lower value should appear in the output first.</p> <p>Constraints:</p> <p>$1 \leq N \leq 10^5$ $1 \leq A[i] \leq 10^4$ $1 \leq K \leq 10^4$</p> <p>Input Format:</p> <p>The first line consists of two space separated integers N and K. The next line consists of N space separated integers denoting the elements of array A.</p> <p>Output Format:</p> <p>Print N space separated integers denoting the elements of the array in the order in which they are required.</p>		
			<p>Logical Test Cases</p>		

```
#include <stdio.h>
#include <stdlib.h>
void count(int a[],int n, int k){
    int *f,*temp,i;
    temp=(int*)malloc(n*sizeof(int));
    f=(int*)calloc(k,sizeof(int));
    for(i=0;i<n;i++)
        f[a[i]%k]++;
    for(i=k-2;i>=0;i--){
        f[i]=f[i]+f[i+1];
    }
    for(i=n-1;i>=0;i--){
        temp[f[a[i]%k]-1]=a[i];
        f[a[i]%k]--;
    }
}
```

```

        for(i=0;i<n;i++)
            printf("%d ",temp[i]);
    }
    void sort(int a[],int n,int k,int m){
        int *temp,*f,i;
        f=(int*)calloc(m+1,sizeof(int));
        temp=(int*)malloc(n*sizeof(int));
        for(i=0;i<n;i++){
            f[a[i]]++;
        }
        for(i=1;i<=m;i++){
            f[i]=f[i]+f[i-1];
        }
        for(i=n-1;i>=0;i--){
            temp[f[a[i]]-1]=a[i];
            f[a[i]]--;
        }
        count(temp,n,k);
    }
}
int main()
{
    int n,k,i,*a,max=0;
    scanf("%d %d",&n,&k);
    a=(int*)malloc(n*sizeof(int));
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
        if(max<a[i])
            max=a[i];
    }
    sort(a,n,k,max);

    return 0;
}

```

CHALLENGE INFORMATION

You have already solved this challenge ! Though you can run the code with different logic !

Course	C	Session	Question Information	Level 1 Challenge 76
Structure Pointers & Array Pointers				
Problem			<p>Problem Description</p> <p>The brave Knight came to the King and asked permission to marry the princess. The King knew that the Knight was brave, but he also wanted to know if he was smart enough. So he asked him to solve the following task.</p> <p>There is a permutation π of numbers from 1 to $2n$. You can make two types of operations.</p> <p>Swap p_1 and p_2, p_3 and p_4, ..., p_{2n-1} and p_{2n}.</p> <p>Swap p_1 and p_{n+1}, p_2 and p_{n+2}, ..., p_n and p_{2n}.</p> <p>The task is to find the minimal number of operations required to sort the given permutation.</p> <p>The Knight was not that smart actually, but quite charming, so the princess asks you to help him to solve the King's task.</p> <p>Constraints</p> <p>$1 \leq n \leq 1000$</p> <p>Input</p> <p>The first line contains the integer n. The second line contains $2n$ integers π_i — the permutation of numbers from 1 to $2n$.</p> <p>Output</p> <p>Print one integer — the minimal number of operations required to sort the permutation. If it is impossible to sort the permutation using these operations, print -1.</p>	
<p>Logical Test Cases</p> <p>Test Case 1 Test Case 2</p>				

```

#include<stdint.h>
#include<stdio.h>
void option1(int *arr,int n){
    int t=0,i;
    for( i=0;i<n;i++){
        t=arr[2*i];
        arr[2*i]=arr[2*i+1];
        arr[2*i+1]=t;
    }
}
void option2(int *arr,int n){
    int t=0,i;
    for( i=0;i<n;i++){
        t=arr[i];
        arr[i]=arr[i+n];
        arr[i+n]=t;
    }
}
int main()
{
    int n,i,j;
    scanf("%d",&n);
    int arr[2*n], arr_2[2*n];
    for( i=0; i < 2*n; i++)
    {
        scanf("%d",&arr[i]);
        arr_2[i] = arr[i];
    }
    int t1=-1,t2=-1;
    for(i=0;i<2*n;i++){

```

```

        if(arr[j]!=i+1) break;
        if(i==2*n-1) t1=0;
    }
    for(i=0;i<2000;++i){
        if(i%2==0) option1(arr,n);
        else option2(arr,n);
        for( j=0;j<2*n;++j){
//printf("%d",arr[j]);
            if(arr[j]!=j+1) break;
            if(j==2*n-1) t1=i+1;
        }
        if(t1!=-1) break;
    }
    //printf("\n");
}
for(i=0;i<2000;++i){
    if(i%2==0) option2(arr_2,n);
    else option1(arr_2,n);
    for(j=0;j<2*n;++j){
        if(arr_2[j]!=j+1) break;
        if(j==2*n-1) t2=i+1;
    }
    if(t2!=-1) break;
}
if(t1<t2) printf("%d\n",t1);
else printf("%d\n",t2);
return 0;
}

```

The screenshot shows a web browser window with the URL `care.srmt.edu.in/srmtretelab/#/srmtretelab/student/home`. The page displays a problem titled "Challenge 77" under the "Question Information" tab. The problem description is as follows:

Problem Description:

An agent called Cypher is decrypting a message, that contains a composite number n . All divisors of n , which are greater than 1, are placed in a circle. Cypher can choose the initial order of numbers in the circle.

In one move Cypher can choose two adjacent numbers in a circle and insert their least common multiple between them. He can do that move as many times as needed.

A message is decrypted, if every two adjacent numbers are not coprime. Note that for such constraints it's always possible to decrypt the message.

Find the minimal number of moves that Cypher should do to decrypt the message, and show the initial order of numbers in the circle for that.

Constraints

$1 \leq t \leq 100$
 $4 \leq n \leq 109$

Input Format:

The first line contains an integer t — the number of test cases. The next t lines describe each test case.

In a single line of each test case description, there is a single composite number n — the number from the message.

It's guaranteed that the total number of divisors of n for all test cases does not exceed $2 \cdot 10^5$.

Output Format:

For each test case in the first line output the initial order of divisors, which are greater than 1, in the circle. In the second line output, a minimal number of moves needed to decrypt the message.

Problem

If there are different possible orders with a correct answer, print any of them.

Explanation:

Consider the least common multiple as follows...

```

#include <stdio.h>
#include <string.h>

#define K 20000

int main() {
    int t;

    scanf("%d", &t);
    while (t--) {
        static int pp[K], dd[K];
        static char used[K];
        int n, n_, kp, kd, p, d, g, h;

        scanf("%d", &n);
        n_ = n;
        kp = 0;
        for (p = 2; p <= n / p; p++)
            if (n % p == 0) {
                while (n % p == 0)
                    n /= p;
                pp[kp++] = p;
            }
        if (n > 1)
            pp[kp++] = n;
        n = n_;
        kd = 0;
        for (d = 2; d <= n / d; d++)
            if (n % d == 0) {
                dd[kd++] = d;
                if (d != n / d)
                    dd[kd++] = n / d;
            }
        if (kp == 2 && pp[0] * pp[1] == n) {
            printf("%d %d %d\n", pp[0], pp[1], n);
            printf("1\n");
        }
    }
}

```

```

        continue;
    }
    memset(used, 0, kd * sizeof *used);
    for (g = 0; g + 1 < kp; g++) {
        int d = pp[g] * pp[g + 1];

        for (h = 0; h < kd; h++)
            if (dd[h] == d) {
                used[h] = 1;
                break;
            }
    }
    for (g = 0; g < kp; g++) {
        p = pp[g];
        for (h = 0; h < kd; h++)
            if (!used[h] && dd[h] % p == 0)
                printf("%d ", dd[h]), used[h] = 1;

        if (g + 1 < kp)
            printf("%d ", pp[g] * pp[g + 1]);
    }
    printf("%d\n", n);
    printf("0\n");
}
return 0;
}

```

You have already solved this challenge! Though you can run the code with different logic!

Course	C	Session	Structure Pointers & Array Pointers	Question Information
				Level 1 • Challenge 77

Problem Description:

An agent called Cypher is decrypting a message, that contains a composite number n . All divisors of n , which are greater than 1, are placed in a circle. Cypher can choose the initial order of numbers in the circle.

In one move Cypher can choose two adjacent numbers in a circle and insert their least common multiple between them. He can do that move as many times as needed.

A message is decrypted, if every two adjacent numbers are not coprime. Note that for such constraints it's always possible to decrypt the message.

Find the minimal number of moves that Cypher should do to decrypt the message, and show the initial order of numbers in the circle for that.

Constraints

1 ≤ n ≤ 100
4 ≤ n ≤ 109

Input Format:

The first line contains an integer t — the number of test cases. The next t lines describe each test case.

In a single line of each test case description, there is a single composite number n — the number from the message.

It's guaranteed that the total number of divisors of n for all test cases does not exceed $2 \cdot 10^5$.

Output Format:

For each test case in the first line output the initial order of divisors, which are greater than 1, in the circle. In the second line output, a minimal number of moves needed to decrypt the message.

Problem

If there are different possible orders with a correct answer, print any of them.

Explanation:

```

#include <stdio.h>
#include <string.h>

#define K 200000

int main() {
    int t;

    scanf("%d", &t);
    while (t--) {
        static int pp[K], dd[K];
        static char used[K];
        int n, n_, kp, kd, p, d, g, h;

        scanf("%d", &n);
        n_ = n;
        kp = 0;
        for (p = 2; p <= n / p; p++)
            if (n % p == 0) {
                while (n % p == 0)
                    n /= p;
                pp[kp++] = p;
            }

        if (n > 1)
            pp[kp++] = n;

        n = n_;
        kd = 0;
        for (d = 2; d <= n / d; d++)
            if (n % d == 0) {
                dd[kd++] = d;
                if (d != n / d)
                    dd[kd++] = n / d;
            }

        if (kp == 2 && pp[0] * pp[1] == n) {
            printf("%d %d %d\n", pp[0], pp[1], n);
            printf("1\n");
            continue;
        }
    }
}

```

```

memset(used, 0, kd * sizeof *used);
for (g = 0; g + 1 < kp; g++) {
    int d = pp[g] * pp[g + 1];

    for (h = 0; h < kd; h++)
        if (dd[h] == d) {
            used[h] = 1;
            break;
        }
}
for (g = 0; g < kp; g++) {
    p = pp[g];
    for (h = 0; h < kd; h++)
        if (!used[h] && dd[h] % p == 0)
            printf("%d ", dd[h]), used[h] = 1;

    if (g + 1 < kp)
        printf("%d ", pp[g] * pp[g + 1]);
}
printf("%d\n", n);
printf("0\n");
}
return 0;
}

```

CHALLENGE INFORMATION

You have already solved this challenge ! Though you can run the code with different logic !

Course	C	Session	Structure Pointers & Array Pointers	Question Information	Level 1 Challenge 78
Problem			<p>Problem Description: Vijay has given a set of points x_1, x_2, \dots, x_n on the number line.</p> <p>Two points i and j can be matched with each other if the following conditions hold: neither i nor j is matched with any other point; $x_i - x_j \geq z$.</p> <p>What is the maximum number of pairs of points you can match with each other?</p> <p>Constraints: $2 \leq n \leq 2 \cdot 10^5$, $1 \leq z \leq 10^9$, $1 \leq x_i \leq 10^9$</p> <p>Input Format: The first line contains two integers n and z — the number of points and the constraint on the distance between matched points, respectively.</p> <p>The second line contains n integers x_1, x_2, \dots, x_n.</p> <p>Output Format: Print the output in a single line contains the maximum number of pairs of points you can match with each other.</p> <p>Explanation: Assume the input as follows: 4 2 1 3 3 7</p>		

```

#include<stdio.h>
#include<stdlib.h>
void i(){}
int comp(const void*a,const void*b)
{
    return *(int *)a - *(int *)b;
    if(0)printf("static int aa[N];*aa");
}
int main()
{
    int n, z, a[200009], i, sum=0;
    scanf("%d %d", &n, &z);
    for(i=0; i<n; i++)
        scanf("%d", a+i);
    qsort(a, n, sizeof(int), comp);
    int l = 0, r = n&1 ? (n>>1)+1 : n>>1;
    for(i=0; i<n; i++)
        while(r < n)
        {
            if(a[r]-a[l] >= z)
                sum++, l++;
            r++;
        }
    printf("%d", sum);
    return 0;
}

```


CHALLENGE INFORMATION

You have already solved this challenge ! Though you can run the code with different logic !

Course	C	Session	Structure Pointers & Array Pointers	Question Information	Level 1 Challenge 79
Problem	<p>Problem Description</p> <p>There is a binary string a of length n. In one operation, you can select any prefix of a with an equal number of 0 and 1 symbols. Then all symbols in the prefix are inverted: each 0 becomes 1 and each 1 becomes 0.</p> <p>For example, suppose $a=0111010000$.</p> <p>In the first operation, we can select the prefix of length 8 since it has four 0's and four 1's: $[01110100]00 \rightarrow [10001011]00$.</p> <p>In the second operation, we can select the prefix of length 2 since it has one 0 and one 1: $[10]000101100 \rightarrow [01]000101100$.</p> <p>It is illegal to select the prefix of length 4 for the third operation, because it has three 0's and one 1.</p> <p>Can you transform the string a into the string b using some finite number of operations (possibly, none)?</p> <p>Constraints</p> <p>$1 \leq t \leq 10^4$</p> <p>$1 \leq n \leq 2 \cdot 10^5$</p> <p>Input Format</p> <p>The first line contains a single integer t — the number of test cases.</p> <p>The first line of each test case contains a single integer n — the length of the strings a and b.</p>				

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
int main() {
    int n_cases, n, balance, diff;
    char s1[300001], s2[300001], *c1, *c2;
    bool any_same, any_different;
    scanf("%d", &n_cases);
    while (n_cases--) {
        scanf("%d", &n);
        scanf("%s\n%s", s1, s2);
        c1 = s1;
        c2 = s2;
        any_same = false;
        any_different = false;
        balance = 0;
        diff = 0;
        while (*c1) {
            any_same = any_same || *c1 == *c2;
            any_different = any_different || *c1 != *c2;
            if (any_same && any_different) break;
            balance += *c2 == '1' ? 1 : -1;
            diff += *c1 - *c2;
            if (balance == 0) {
                any_same = false;
                any_different = false;
            }
            c1++;
            c2++;
        }
        printf(((any_same && any_different) || diff != 0) ? "NO\n" : "YES\n");
    }
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
int cmp(const void *a, const void *b) {
    return *(int*)a - *(int*)b;
}
int main() {
    int o[2000], ol = 0, e[2000], el = 0, n, t;
    scanf("%d", &n);
    while(n--) {
        scanf("%d", &t);
        if(t % 2)
            o[ol++] = t;
        else
            e[el++] = t;
    }
    qsort(o, ol, sizeof(int), cmp);
    qsort(e, el, sizeof(int), cmp);
    while(ol && el) {
        ol--;
        el--;
    }
    t = 0;
    if(ol) {
        ol--;
        while(ol)
            t += o[--ol];
    } else if(el) {
        el--;
        while(el)
            t += e[--el];
    }
    printf("%d", t);
    return 0;
}
```