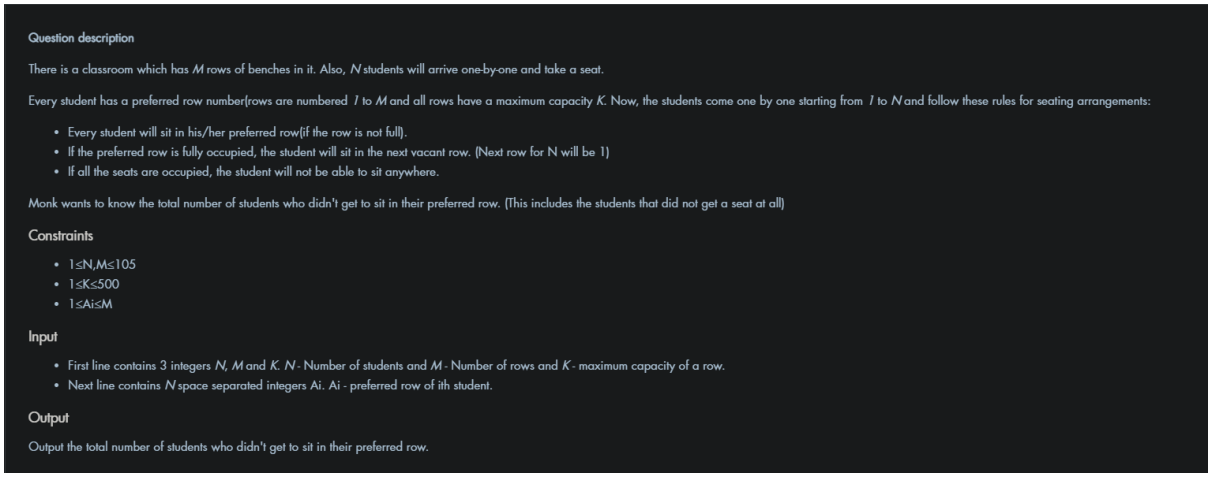
Data Structures and Algorithms (18CSC201J) - RA2111032010022



Question description

There is a classroom which has *M* rows of benches in it. Also, *N* students will arrive one-by-one and take a seat.

Every student has a preferred row number(rows are numbered *1* to *M* and all rows have a maximum capacity *K*. Now, the students come one by one starting from *1* to *N* and follow these rules for seating arrangements:

- Every student will sit in his/her preferred row(if the row is not full).
- If the preferred row is fully occupied, the student will sit in the next vacant row. (Next row for N will be 1)
- If all the seats are occupied, the student will not be able to sit anywhere.

Monk wants to know the total number of students who didn't get to sit in their preferred row. (This includes the students that did not get a seat at all)

Constraints

- 1≤N,M≤105
- 1≤K≤500
- 1≤Ai≤M

Input

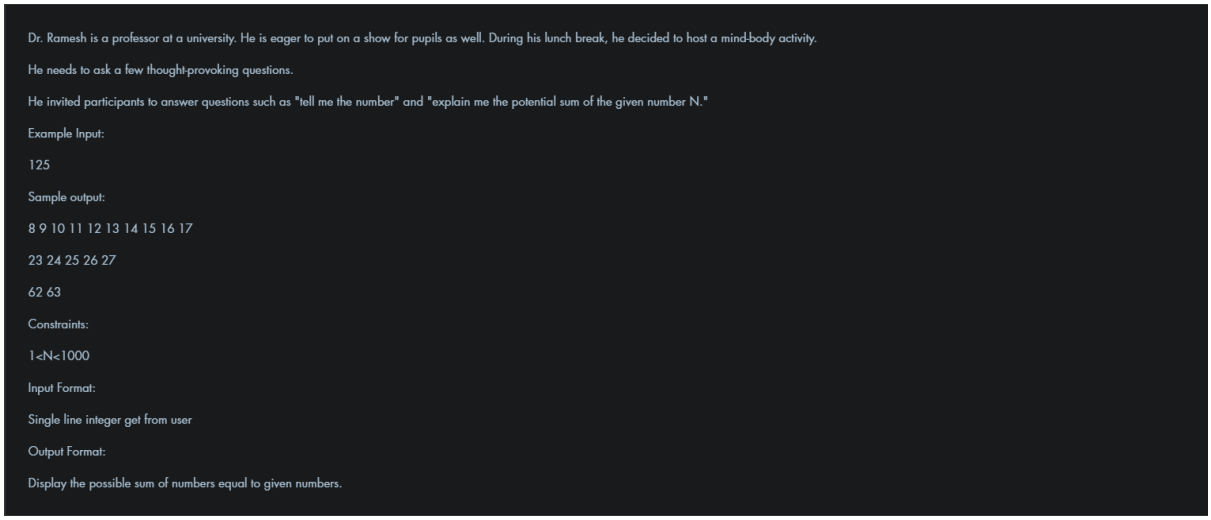
- First line contains 3 integers *N*, *M* and *K*. *N* - Number of students and *M* - Number of rows and *K* - maximum capacity of a row.
- Next line contains *N* space separated integers Ai. Ai - preferred row of ith student.

Output

Output the total number of students who didn't get to sit in their preferred row.

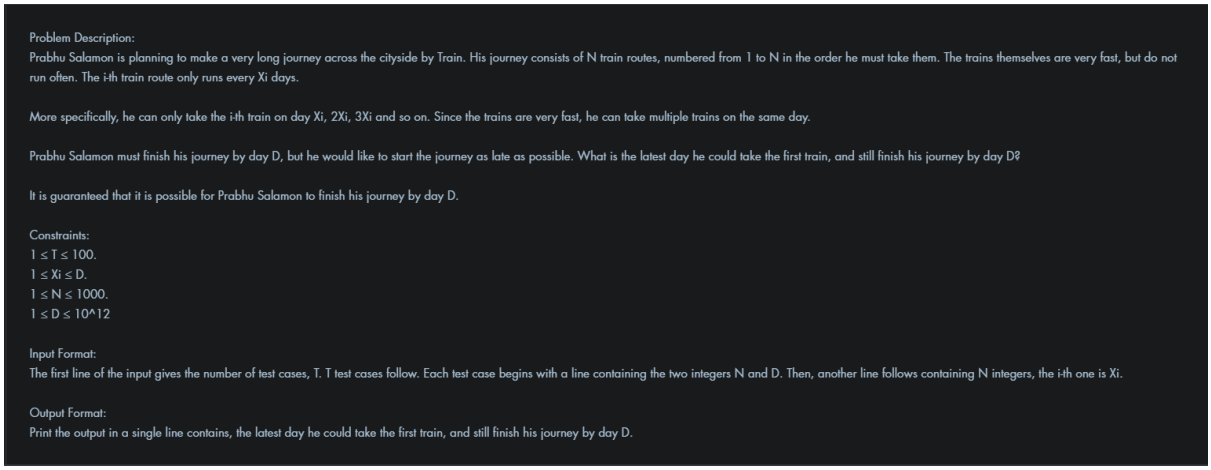
```
#include <stdio.h>
#include <malloc.h>

int main()
{
    int n, m, k, x, y, i, *a, *b, ans = 0, flag = 1;
    scanf("%d %d %d", &n, &m, &k);
    a = (int *)calloc(n, sizeof(int));
    b = (int *)calloc(n, sizeof(int));
    for (i = 0; i < n; i++)
    {
        scanf("%d", &x);
        if (a[x] < k)
        {
            ans++;
            a[x]++;
        }
        else if (flag != 0)
        {
            y = x;
            x++;
            if (b[y] != 0)
                x = b[y];
            flag = 0;
            while (x != y)
            {
                if (x == m + 1)
                    x = 1;
```

```
                if (x == y)
                    break;
                if (a[x] < k)
                {
                    a[x]++;
                    flag = 1;
                    b[y] = x;
                    break;
                }
                x++;
            }
        }
    }
    printf("%d", n - ans);
    return 0;
}
```




```
#include <iostream>
using namespace std;

void printNumbers(int start, int stop)
{
    int i;
    for (i = start; i <= stop; i++)
    {
        cout << i << " ";
    }
    cout << endl;
}

void printSums(int N)
{
    int i;
    for (i = 1; i < N; i++)
    {
        int start = i, stop = i, j, sum = 0;
        for (j = i; i < N; j++)
```

```
        {
            sum += j;
            stop = j;
            if (sum == N)
            {
                printNumbers(start, stop);
                break;
            }
            else if (sum > N)
            {
                break;
            }
        }
    }
}

int main()
{
    int n;
    cin >> n;
    printSums(n);
    return 0;
}
```



Problem Description:
Prabhu Salamon is planning to make a very long journey across the cityside by Train. His journey consists of N train routes, numbered from 1 to N in the order he must take them. The trains themselves are very fast, but do not run often. The i-th train route only runs every Xi days.

More specifically, he can only take the i-th train on day Xi, 2Xi, 3Xi and so on. Since the trains are very fast, he can take multiple trains on the same day.

Prabhu Salamon must finish his journey by day D, but he would like to start the journey as late as possible. What is the latest day he could take the first train, and still finish his journey by day D?

It is guaranteed that it is possible for Prabhu Salamon to finish his journey by day D.

Constraints:
1 ≤ T ≤ 100.
1 ≤ Xi ≤ D.
1 ≤ N ≤ 1000.
1 ≤ D ≤ 10^12

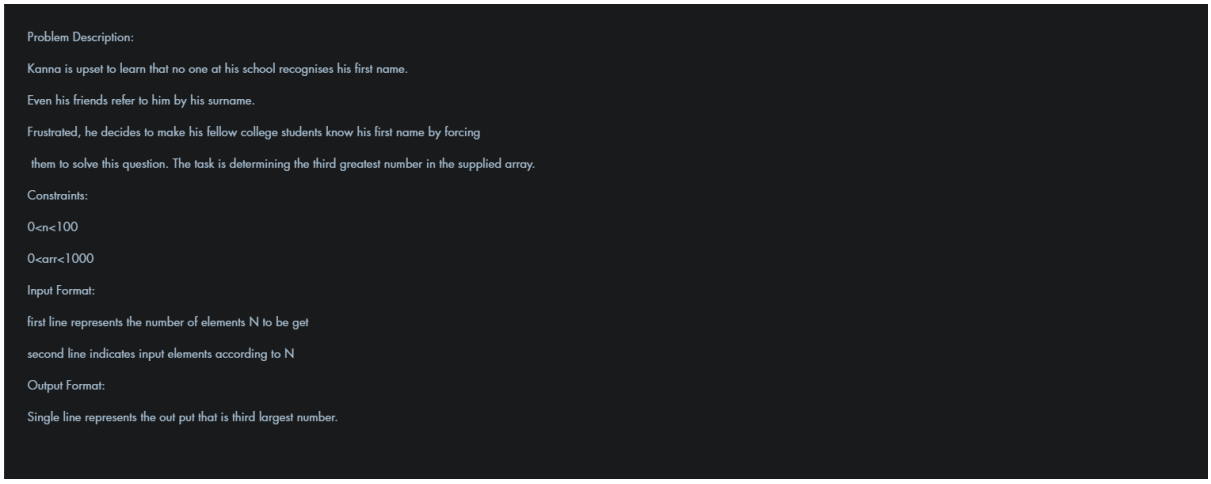
Input Format:
The first line of the input gives the number of test cases, T. T test cases follow. Each test case begins with a line containing the two integers N and D. Then, another line follows containing N integers, the i-th one is Xi.

Output Format:
Print the output in a single line contains, the latest day he could take the first train, and still finish his journey by day D.

```
#include <stdio.h>
#include <malloc.h>

int main()
{
    int T;
    scanf("%d", &T);
    for (int t = 0; t < T; t++)
    {
        int n, d, *days;
        scanf("%d %d", &n, &d);
        days = (int *)malloc(sizeof(int) * n);
        for (int i = 0; i < n; i++)
        {
            scanf("%d", &days[i]);
```

```
        }
        for (int i = n - 1; i >= 0; i--)
        {
            d -= d % days[i];
        }
        printf("%d\n", d);
    }
    return 0;
}
```


```
#include <iostream>
using namespace std;

int main()
{
    int t, count = 0;
    cin >> t;
    while (t--)
    {
        double width, height;
        cin >> width >> height;
        if (width / height >= 1.6 && width / height <= 1.7)
            count++;
        else if (height / width >= 1.6 && height / width <= 1.7)
            count++;
    }
    cout << count;
    return 0;
}
```




```
#include <stdio.h>
#include <malloc.h>

void thirdLargest(int arr[], int arr_size)
{
    int i, j;
    for (i = 0; i < arr_size; i++)
    {
        for (j = 0; j < i; j++)
        {
            if (arr[i] > arr[j])
            {
                int tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
            }
        }
    }
    printf("The third Largest element is %d", arr[2]);
}

int main()
{
    int arr_size, i;
    scanf("%d", &arr_size);
    int arr[arr_size];
    for (i = 0; i < arr_size; i++)
    {
        scanf("%d", &arr[i]);
    }
    thirdLargest(arr, arr_size);
    return 0;
}
```

Question description

Nathan won the man of the match award in the recently concluded local tournament final.

So the friends of nathan have asked him to take them to cinemas as a trat for winning man of the match.But Nathan is short of money to take them to cinemas so to postpone the cinema plan he tred to engage them with the programming challenge.

The task is, Given an N-dimensional array Arr, where all elements are bigger than or equal to zero. Return the highest possible product of two numbers.

There's no need to read or print anything. Your objective is to complete the procedure maxProduct, which accepts as parameters an array of integers Arr[] and n and returns an integer indicating the answer.

Constraints:
$2 \leq N \leq 10^7$
$0 \leq Arr[i] \leq 10^4$

Input Format:

The first line of input contains an integer T denoting the number of test cases.

The first line of each test case is N, N is size of array.

The second line of each test case contains N input Arr[i].

Output Format:

Single line represents the maximum product of two numbers possible.

```
#include <stdio.h>
#include <malloc.h>

void sort(int a[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - 1 - i; j++)
            if (a[j] < a[j + 1])
            {
                int tmp = a[j + 1];
                a[j + 1] = a[j];
                a[j] = tmp;
            }
}

int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
    {
        int n, i, *arr;
        scanf("%d", &n);
        arr = (int *)malloc(sizeof(int) * n);
        for (i = 0; i < n; i++)
            scanf("%d", &arr[i]);
        sort(arr, n);
        printf("%d\n", arr[0] * arr[1]);
    }
    return 0;
}
```

Question description

In India, the real estate sector is the second-highest employment generator, after the agriculture sector.

It is also expected that this sector will incur more non-resident Indian (NRI) investment, both in the short term and the long term.

Bengaluru is expected to be the most favoured property investment destination for NRIs, followed by Ahmedabad, Pune, Chennai, Goa, Delhi and Dehradun.

Ramesh is residing in England. he is willing to invest money in real estate.

So he has chosen Bengaluru for good investment.
There are N flats for sale in Bengaluru main city.

The i-th flat costs Ai rupees to buy.

Ramesh has a budget of B rupees to spend.

What is the maximum number of flats Ramesh can buy?

Constraints:
1 ≤ T ≤ 100.
1 ≤ B ≤ 10^5.
1 ≤ Ai ≤ 1000, for all i.
1 ≤ N ≤ 10^5.

Input Format:
The first line of the input gives the number of test cases, T.

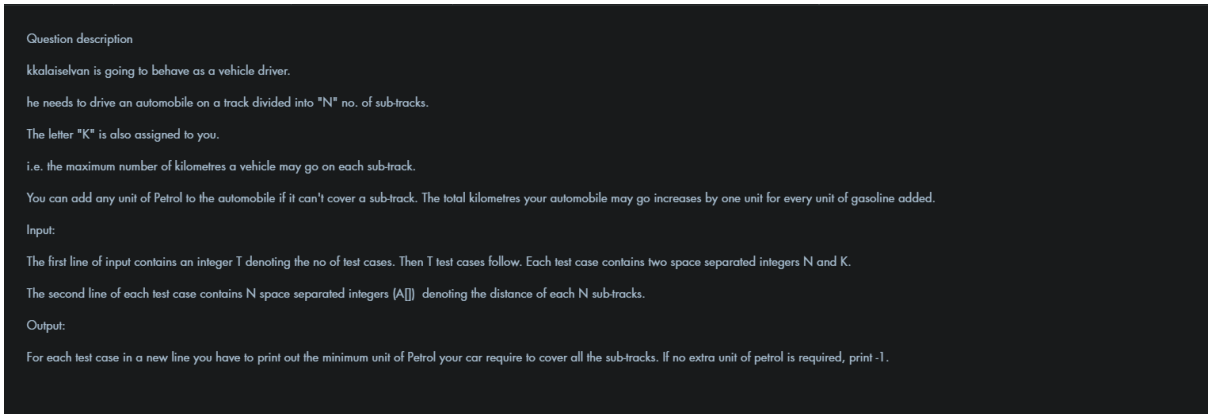
T test cases follow. Each test case begins with a single line containing the two integers N and B.

The second line contains N integers. The i-th integer is Ai, the cost of the i-th flat.

Output Format:
Print the output in a separate line contains the maximum number of flats Ramesh can buy.

```
#include <stdio.h>
#include <malloc.h>

void makeheap(int x[], int n)
{
    int i;
    for (i = 0; i < n; i++)
    {
        scanf("%d", &x[i]);
    }
}

void heapsort(int x[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)
    {
        for (j = 0; j < n - i - 1; j++)
        {
            if (x[j] > x[j + 1])
            {
                int tmp = x[j + 1];
                x[j + 1] = x[j];
                x[j] = tmp;
            }
        }
    }
}

int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
```

```
    {
        int n, b, i, sum = 0, count = 0, *a;
        scanf("%d %d", &n, &b);
        a = (int *)malloc(sizeof(int) * n);
        makeheap(a, n);
        heapsort(a, n);
        for (i = 0; i < n; i++)
        {
            sum += a[i];
            if (sum <= b)
                count++;
        }
        printf("%d\n", count);
    }
    return 0;
}
```




```
#include <stdio.h>
#include <malloc.h>

void sort(int a[],int n)
{
    int i, j;
    for(i=0;i<n-1;i++)
    {
        for (j = i; j < n; j++)
        {
            if (a[i] < a[j])
            {
                int tmp = a[i];
                a[i] = a[j];
                a[j] = tmp;
            }
        }
    }
}

int main()
{
    int t;
```

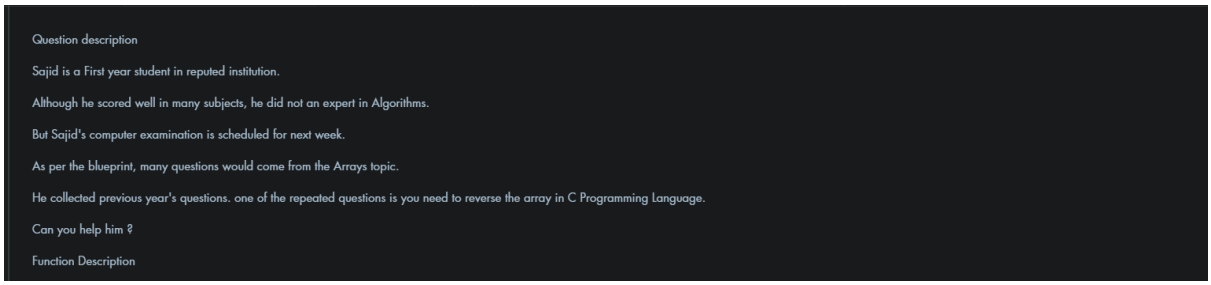
```
    scanf("%d", &t);
    while(t--)
    {
        int n, k, i, *arr;
        scanf("%d %d", &n, &k);
        arr = (int *)malloc(sizeof(int) * n);
        for (i = 0; i < n; i++)
        {
            scanf("%d", &arr[i]);
        }
        sort(arr, n);
        printf("%d\n", arr[0] - k > 0 ? arr[0] - k : -1);
    }
    return 0;
}
```


```
#include <iostream>
#include <algorithm>
using namespace std;

void sort(int a[], int n, int flag);

class Sort
{
public:
    int a[100], b[100];
    int n;
    void getn()
    {
        cin >> n;
    }
    void geta()
    {
        for (int i = 0; i < n; i++)
            cin >> a[i];
```

```
        sort(a, a + n);
    }
    void getb()
    {
        for (int i = 0; i < n; i++)
            cin >> b[i];
        sort(b, b + n);
    }
    void display()
    {
        int sum = 0;
        for (int i = 0; i < n; i++)
            sum += a[i] * b[n - i - 1];
        cout << sum << endl;
    }
};
int main()
{
    int n;
    cin >> n;
    while (n--)
    {
        Sort t;
        t.getn();
        t.geta();
        t.getb();
        t.display();
    }
    return 0;
}
```


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

int n, i, j, sum = 0, arr[100000];

void swap(int l, int r)
{
```

```
    arr[i] = r;
    arr[j] = l;
}

int main()
{
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    for (i = 0; i < n; i++)
    {
        for (j = i; j < n; j++)
        {
            int l = arr[i], r = arr[j];
            if (arr[i] < arr[j])
                swap(l, r);
        }
        sum += abs(arr[i] - i - 1);
    }
    printf("%d", sum);
    return 0;
}
```




```
#include <iostream>
#include <malloc.h>
using namespace std;

int main()
{
    int n, i, temp;
    cin >> n;
    int arr[n];
    for (i = 0; i < n; i++)
        cin >> arr[i];
    for (i = 0; i < n / 2; i++)
    {
        temp = arr[i];
        arr[i] = arr[n - i - 1];
        arr[n - i - 1] = temp;
    }
    for (i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Question description

Malar is a First year student in reputed institution.

Although he scored well in many subjects, he did not an expert in Algorithms.

But malar's computer examination is scheduled for next week.

As per the blueprint, many questions would come from the Arrays topic.

He collected previous year's questions. one of the repeated questions is you need to find the pairs in Array with given sum.

Can you help him ?

Function Description

Sample Test Case

```
#include <iostream>
using namespace std;

int main()
{
    int n, sum = 0, count = 0, i = 0, j = 0;
    cin >> n;
    int array[n];
    for (i = 0; i < n; i++)
        cin >> array[i];
    cin >> sum;
    for (i = 0; i < n; i++)
    {
        for (j = i + 1; j < n; j++)
        {
            if (array[i] + array[j] == sum)
            {
                count += 1;
                cout << "[" << array[i] << " " << array[j] << "]" << endl;
            }
        }
    }
    cout << "Total Number of Pairs:" << count;
}
```

Problem Description:
Caleb likes to challenge Selvan's math ability.

He will provide a starting and ending value that describes a range of integers, inclusive of the endpoints.

Selvan must determine the number of square integers within that range.

Note:

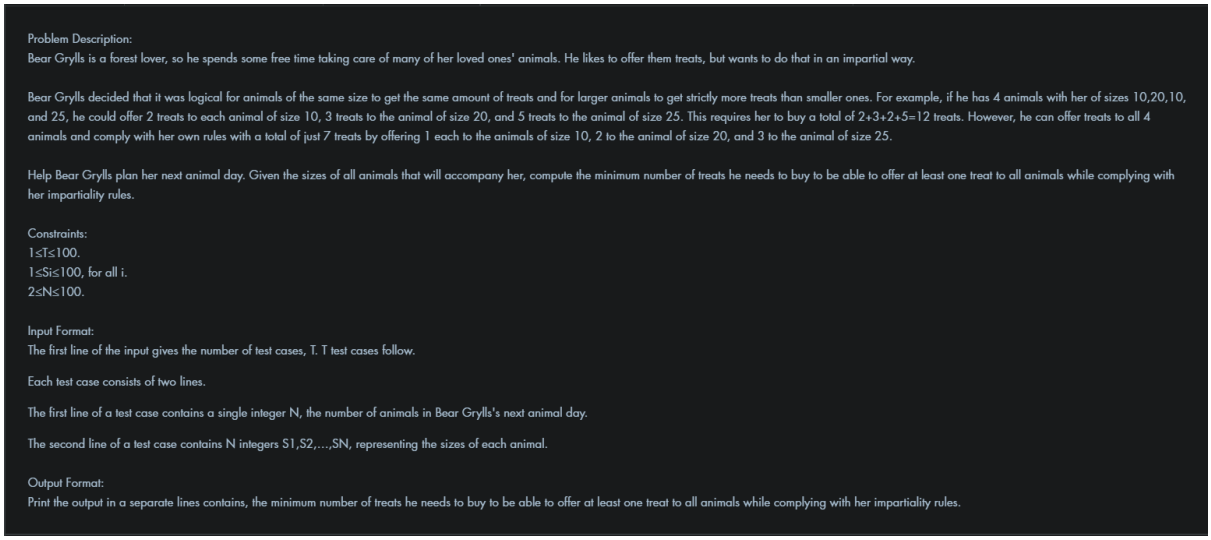
A square integer is an integer which is the square of an integer, e.g. 1, 4, 9, 16, 25

Constraints:
1 <= q <= 100
1 <= start <= 10^9
1 <= end <= 10^9

Input Format
The first line contains 'q', number of test cases.
Each of the next 'q' lines contains two space-separated integers, representing 'start' and 'end'.

Output Format:
Print the number of square integers within that range.

```
#include <iostream>
#include <math.h>
using namespace std;

int main()
{
    int q, number_of_squares;
    cin >> q;
    while (q--)
    {
        int start, stop;
        cin >> start >> stop;
        number_of_squares = floor(sqrt(stop)) - ceil(sqrt(start)) + 1;
        cout << number_of_squares << endl;
    }
    return 0;
}
```



Problem Description:
Bear Grylls is a forest lover, so he spends some free time taking care of many of her loved ones' animals. He likes to offer them treats, but wants to do that in an impartial way.

Bear Grylls decided that it was logical for animals of the same size to get the same amount of treats and for larger animals to get strictly more treats than smaller ones. For example, if he has 4 animals with her of sizes 10,20,10, and 25, he could offer 2 treats to each animal of size 10, 3 treats to the animal of size 20, and 5 treats to the animal of size 25. This requires her to buy a total of 2+3+2+5=12 treats. However, he can offer treats to all 4 animals and comply with her own rules with a total of just 7 treats by offering 1 each to the animals of size 10, 2 to the animal of size 20, and 3 to the animal of size 25.

Help Bear Grylls plan her next animal day. Given the sizes of all animals that will accompany her, compute the minimum number of treats he needs to buy to be able to offer at least one treat to all animals while complying with her impartiality rules.

Constraints:
1≤T≤100.
1≤Si≤100, for all i.
2≤N≤100.

Input Format:
The first line of the input gives the number of test cases, T. T test cases follow.

Each test case consists of two lines.

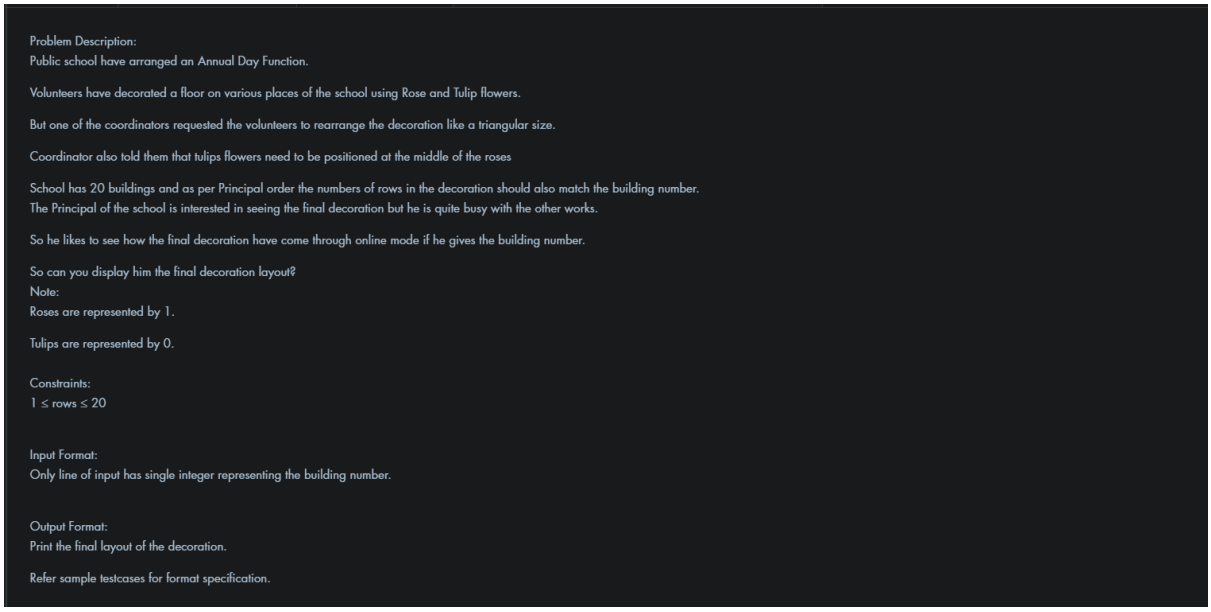
The first line of a test case contains a single integer N, the number of animals in Bear Grylls's next animal day.

The second line of a test case contains N integers S1,S2,...,SN, representing the sizes of each animal.

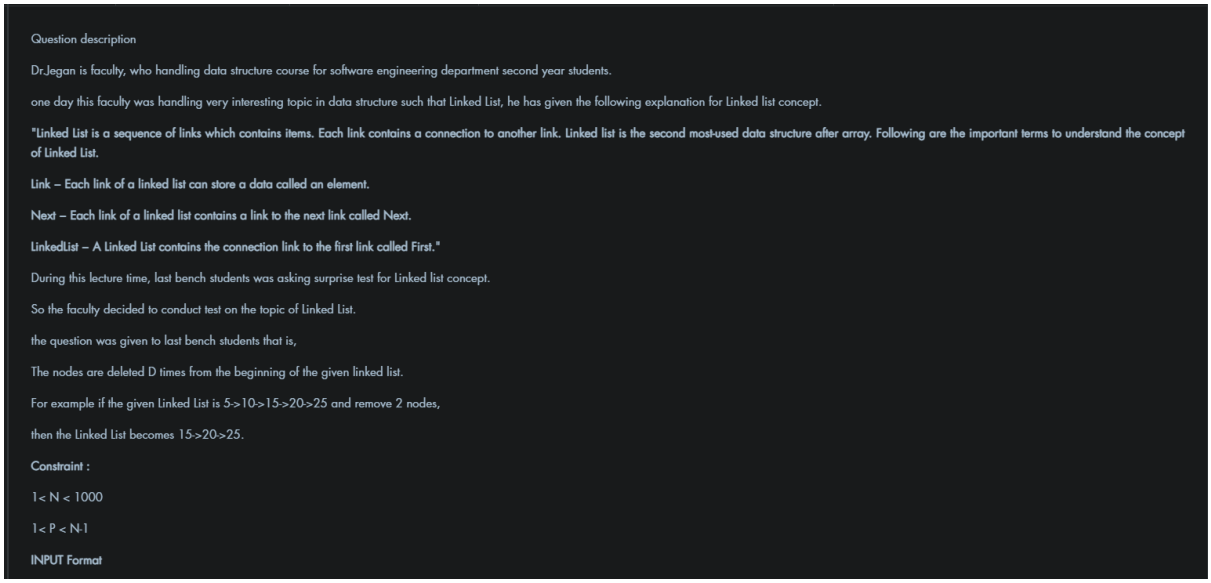
Output Format:
Print the output in a separate lines contains, the minimum number of treats he needs to buy to be able to offer at least one treat to all animals while complying with her impartiality rules.

```
#include <iostream>
#define MAXN 100
using namespace std;

void sort(int n, int *arr)
{
    int i, j;
    for (i = 0; i < n; i++)
    {
        for (j = i; j < n; j++)
        {
            if (arr[i] > arr[j])
            {
                int tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
            }
```

```
        }
    }
}

int read(int s)
{
    cin >> s;
    return s;
}

void sol()
{
    int n, i;
    cin >> n;
    int s[MAXN];
    for (i = 0; i < n; i++)
        s[i] = read(s[i]);
    sort(n, s);
    int total_treats = 0, number_of_treats = 1, largest = s[0];
    for (i = 0; i < n; i++)
    {
        if (s[i] > largest)
        {
            largest = s[i];
            number_of_treats++;
        }
        total_treats += number_of_treats;
    }
    cout << total_treats << endl;
}

int main()
{
    int t;
    cin >> t;
    while (t--)
        sol();
    return 0;
}
```




```
#include <stdio.h>

int main()
{
    int rows, i, j;
    scanf("%d", &rows);
    for (i = 1; i <= rows; i++)
    {
        for (j = 1; j <= rows; j++)
        {
            if (i == rows || j == 1 || j == i)
                printf("1 ");
            else if (i > j)
                printf("0 ");
        }
        printf("\n");
    }
    return 0;
}
```



Question description

Dr.Jegan is faculty, who handling data structure course for software engineering department second year students.

one day this faculty was handling very interesting topic in data structure such that Linked List, he has given the following explanation for Linked list concept.

"Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

Link – Each link of a linked list can store a data called an element.

Next – Each link of a linked list contains a link to the next link called Next.

LinkedList – A Linked List contains the connection link to the first link called First."

During this lecture time, last bench students was asking surprise test for Linked list concept.

So the faculty decided to conduct test on the topic of Linked List.

the question was given to last bench students that is,

The nodes are deleted D times from the beginning of the given linked list.

For example if the given Linked List is 5->10->15->20->25 and remove 2 nodes,

then the Linked List becomes 15->20->25.

Constraint :

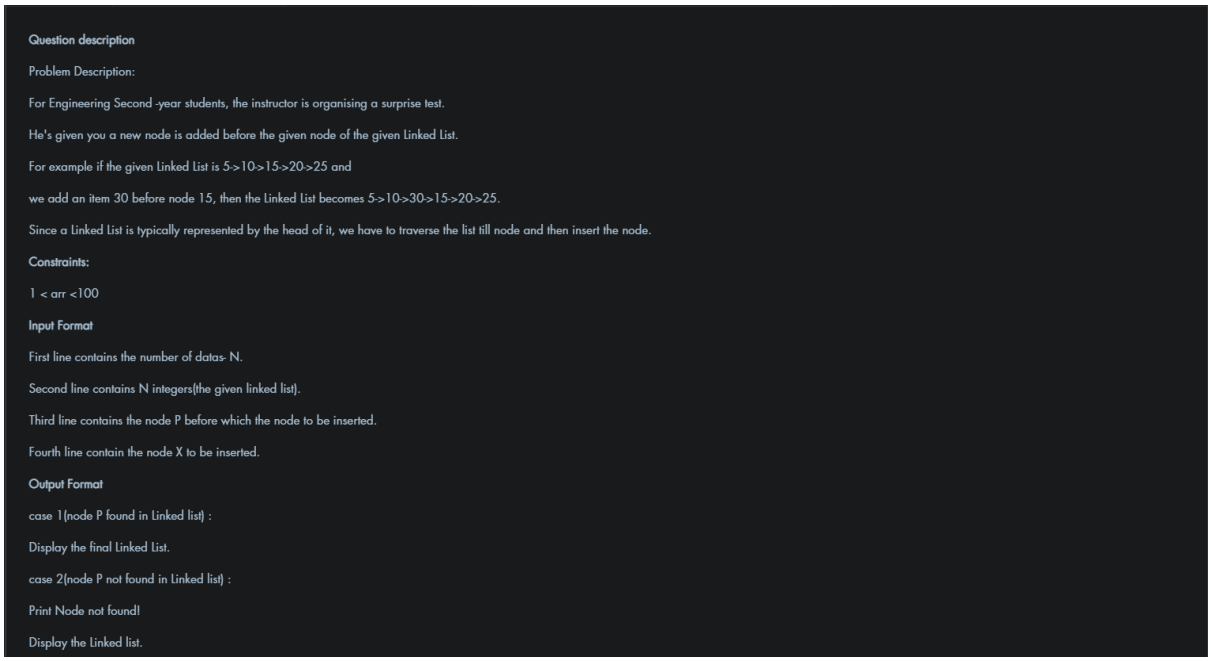
1< N < 1000

1< P < N-1

INPUT Format

```
#include <stdio.h>
#include <malloc.h>

struct node
{
    int data;
    struct node *next;
};

struct node *start = NULL, *ptr, *new_node;
struct node *ptr;
int n, i;

void create()
{
    scanf("%d", &n);

    new_node = malloc(sizeof(struct node));

    scanf("%d", &new_node->data);
    new_node->next = NULL;
    start = new_node;
    n--;
    while (n--)
    {
        new_node = malloc(sizeof(struct node));

        scanf("%d", &new_node->data);
        new_node->next = NULL;

        ptr = start;
        while (ptr->next != NULL)
            ptr = ptr->next;
        ptr->next = new_node;
    }
}
```

```
int main()
{
    create();

    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        ptr = start;
        start = ptr->next;
        free(ptr);
    }

    ptr = start;
    printf("Linked List:");
    while (ptr != NULL)
    {
        printf("->%d", ptr->data);
        ptr = ptr->next;
    }
    return 0;
}
```



Question description

Problem Description:

For Engineering Second -year students, the instructor is organising a surprise test.

He's given you a new node is added before the given node of the given Linked List.

For example if the given Linked List is 5->10->15->20->25 and

we add an item 30 before node 15, then the Linked List becomes 5->10->30->15->20->25.

Since a Linked List is typically represented by the head of it, we have to traverse the list till node and then insert the node.

Constraints:

1 < arr <100

Input Format

First line contains the number of datas- N.

Second line contains N integers(the given linked list).

Third line contains the node P before which the node to be inserted.

Fourth line contain the node X to be inserted.

Output Format

case 1(node P found in Linked list) :

Display the final Linked List.

case 2(node P not found in Linked list) :

Print Node not found!

Display the Linked list.

```
#include <stdio.h>
#include <malloc.h>

struct node
{
    int data;
    struct node *next;
};

struct node *start = NULL, *p1, *pp1, *p2;
int n, data, key;
```

```
struct node *create_node()
{
    p2 = malloc(sizeof(struct node));
    scanf("%d", &p2->data);

    return p2;
}

void create_ll()
{
    scanf("%d", &n);
    while (n--)
    {
        p2 = create_node();

        if (start == NULL)
            start = p2;

        else
        {
            p1 = start;
            while (p1->next != NULL)
                p1 = p1->next;
            p1->next = p2;
        }
    }
}

void insert_before()
{
    scanf("%d", &key);
    p2 = create_node();

    p1 = start;
    while (p1->data != key && p1->next != NULL)
    {
        pp1 = p1;
        p1 = p1->next;
    }

    if (p1->data == key)
    {
        pp1->next = p2;
        p2->next = p1;
    }

    else
        printf("Node not found!\n");
}

void display()
{
    p1 = start;
    printf("Linked List:");
    while (p1 != NULL)
    {
        printf("->%d", p1->data);
```

```
        p1 = p1->next;
    }
}

int main()
{
    create_ll();
    insert_before();
    display();
    return 0;
}
```

Question description

sanam's Dream came true after he got an Appointment order from Google.Simon's family was very happy of his achievement.

The company mentioned Basic Salary, DA, HRA with some other benefits.

But not highlighted the Gross salary in the order.

sanam's father wanted to know the Gross salary of his son.

sanam try to his gross salary from HR department. they informed that you have to get pass grade in first month entry test. the entry test has 5 questions. one of the question was, Split a circular linked list in two halves. you have to split the circular Linked List with the same size of Divisions.Maybe if circular Linked List is odd, you have to change the number of node, it is even .

Can you help sanam?

Function Description

First count the number of node in Circular Linked List.

Second, you have to make the List even.

Third, you need to make the List half. the front is the same size like the rear. Finally, I have to make two circular Linked List.

Constraints

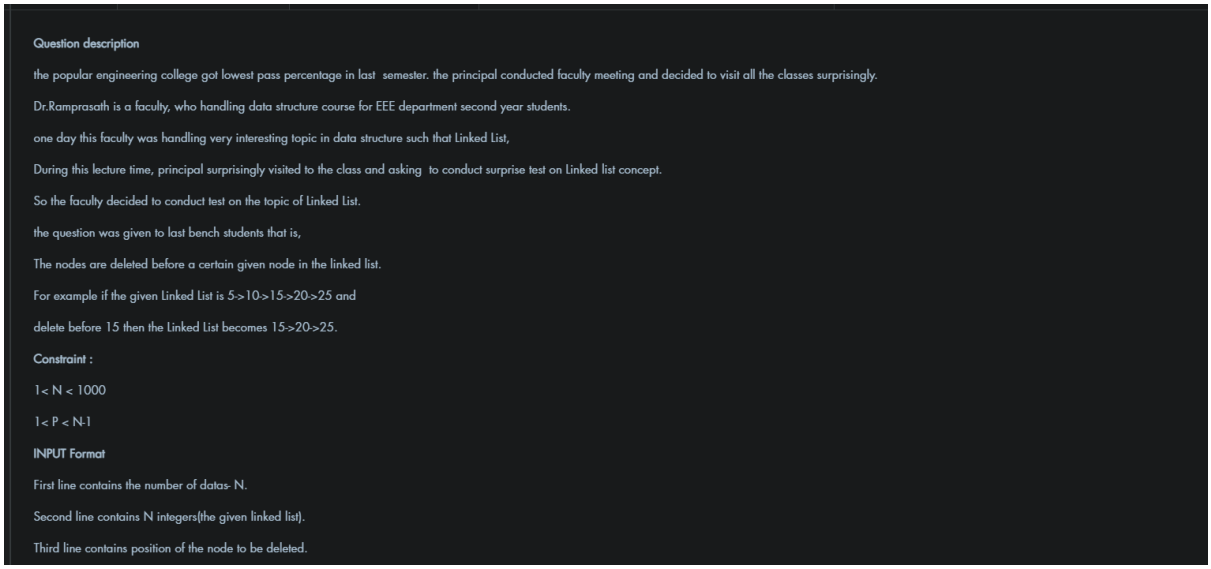
0<n<10

Input Format:
The First line represents the number of input in the circular linked list elements

Output Format:
First Line indicates the complete linked list

```
#include <stdio.h>
#include <malloc.h>

struct n
{
    int data;
    struct n *next;
};

struct n *h = NULL;
void insert(int data);
void display(struct n *h);

int main()
{
    int i, n;
    scanf("%d", &n);
    printf("Complete linked_list:\n");
    for (i = 1; i <= n; i++)
    {
        insert(i);
    }
    display(h);
```

```
    h = NULL;
    printf("Odd:\n");
    for (i = 1; i <= n; i += 2)
    {
        insert(i);
    }
    display(h);

    h = NULL;
    printf("Even:\n");
    for (i = 2; i <= n; i += 2)
    {
        insert(i);
    }
    display(h);
    return 0;
}

void insert(int data)
{
    struct n *new_node, *ptr;

    new_node = (struct n *)malloc(sizeof(struct n));
    new_node->data = data;
    new_node->next = NULL;

    if (h == NULL)
    {
        h = new_node;
    }

    else
    {
        ptr = h;
        while (ptr->next != NULL)
            ptr = ptr->next;
        ptr->next = new_node;
    }
}

void display(struct n *h)
{
    struct n *ptr;
    ptr = h;
    printf("[h]=>");
    while (ptr != NULL)
    {
        printf("%d=>", ptr->data);
        ptr = ptr->next;
    }
    printf("[h]\n");
}
```



Question description

the popular engineering college got lowest pass percentage in last semester. the principal conducted faculty meeting and decided to visit all the classes surprisingly.

Dr.Ramprasath is a faculty, who handling data structure course for EEE department second year students.

one day this faculty was handling very interesting topic in data structure such that Linked List,

During this lecture time, principal surprisingly visited to the class and asking to conduct surprise test on Linked list concept.

So the faculty decided to conduct test on the topic of Linked List.

the question was given to last bench students that is,

The nodes are deleted before a certain given node in the linked list.

For example if the given Linked List is 5->10->15->20->25 and

delete before 15 then the Linked List becomes 15->20->25.

Constraint :

1< N < 1000

1< P < N-1

INPUT Format

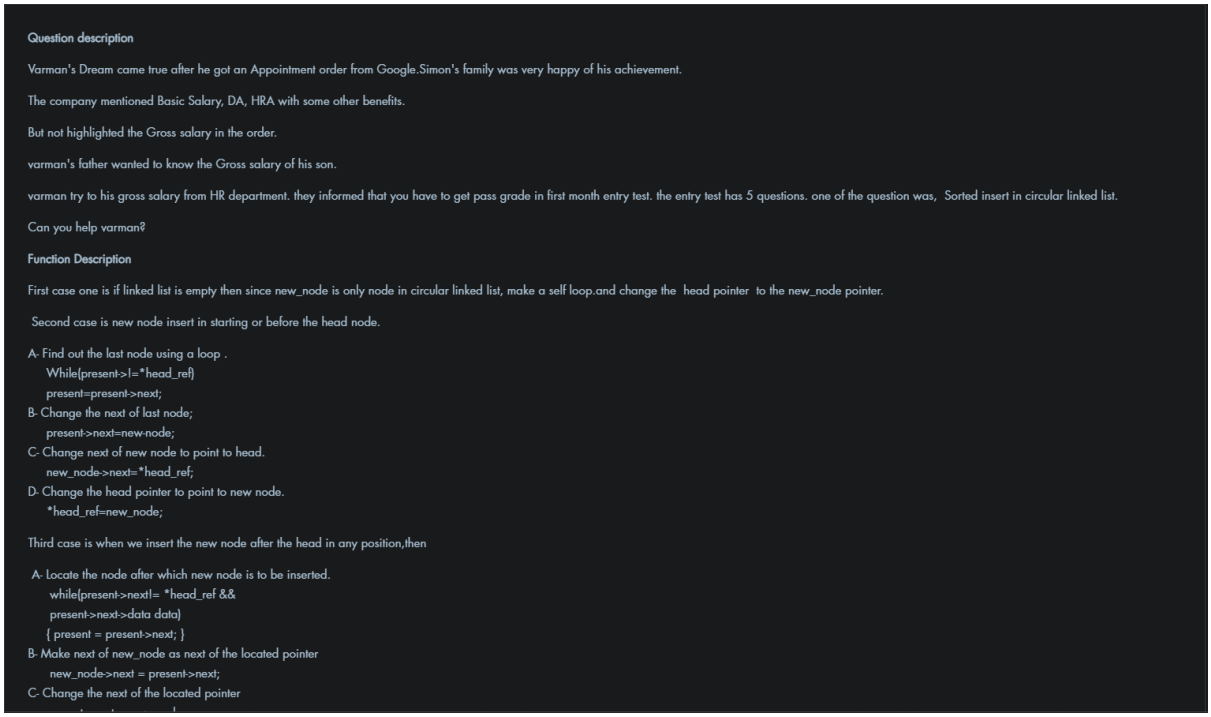
First line contains the number of datas- N.

Second line contains N integers(the given linked list).

Third line contains position of the node to be deleted.

```
#include <stdio.h>
#include <malloc.h>

struct node
{
    int data;
    struct node *next;
};

struct node *head = NULL, *ptr, *p1, *new_node;
int i, n, data;

struct node *create_node()
{
    scanf("%d", &data);
    new_node = malloc(sizeof(struct node));
    new_node->data = data;
    new_node->next = NULL;

    return new_node;
}

void create()
{
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        new_node = create_node();

        if (head == NULL)
            head = new_node;

        else
        {
            p1 = head;
            while (p1->next != NULL)
                p1 = p1->next;
```

```
            p1->next = new_node;
        }
    }
}

void del()
{
    scanf("%d", &data);

    p1 = head;
    while (p1->data != data && p1->next != NULL)
        p1 = p1->next;

    if (p1->data == data)
        head = p1;

    else
        printf("Invalid Node! ");
}

void display()
{
    p1 = head;
    printf("Linked List:");
    while (p1 != NULL)
    {
        printf("->%d", p1->data);
        p1 = p1->next;
    }
}

int main()
{
    create();
    del();
    display();
    return 0;
}
```



Question description

Varman's Dream came true after he got an Appointment order from Google.Simon's family was very happy of his achievement.

The company mentioned Basic Salary, DA, HRA with some other benefits.

But not highlighted the Gross salary in the order.

varman's father wanted to know the Gross salary of his son.

varman try to his gross salary from HR department. they informed that you have to get pass grade in first month entry test. the entry test has 5 questions. one of the question was, Sorted insert in circular linked list.

Can you help varman?

Function Description

First case one is if linked list is empty then since new_node is only node in circular linked list, make a self loop.and change the head pointer to the new_node pointer.

Second case is new node insert in starting or before the head node.

A- Find out the last node using a loop .
While(present->I=*head_ref)
present=present->next;
B- Change the next of last node;
present->next=new-node;
C- Change next of new node to point to head.
new_node->next=*head_ref;
D- Change the head pointer to point to new node.
*head_ref=new_node;

Third case is when we insert the new node after the head in any position,then

A- Locate the node after which new node is to be inserted.
while(present->next!= *head_ref &&
present->next->data data)
{ present = present->next; }
B- Make next of new_node as next of the located pointer
new_node->next = present->next;
C- Change the next of the located pointer

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

struct Node
{
    int data;
    struct Node *next;
};

void sortedInsert(struct Node **head_ref, struct Node *new_node)
{
    struct Node *current = *head_ref;

    if (current == NULL)
    {
        new_node->next = new_node;
        *head_ref = new_node;
    }

    else if (current->data >= new_node->data)
    {
        while (current->next != *head_ref)
            current = current->next;

        current->next = new_node;
        new_node->next = *head_ref;
        *head_ref = new_node;
    }

    else
    {
        while (current->next != *head_ref && current->next->data < new_node->data)
            current = current->next;
```

```
        new_node->next = current->next;
        current->next = new_node;
    }
}

void print(struct Node *start)
{
    struct Node *temp;
    temp = start;

    do
    {
        printf("%d ", temp->data);
        temp = temp->next;
    } while (temp->next != start);
    printf("%d", temp->data);
}

int main()
{
    int n, i;
    scanf("%d", &n);

    struct Node *start = NULL;
    struct Node *temp;

    for (i = 0; i < n; i++)
    {
        temp = (struct Node *)malloc(sizeof(struct Node));
        scanf("%d", &temp->data);
        sortedInsert(&start, temp);
    }
    print(start);
    return 0;
}
```

Question description

Hassan gets a job in a software company in Hyderabad. The training period for the first three months is 20000 salary. Then incremented to 25000 salaries.

Training is great but they will give you a programming task every day in three months. Hassan must finish it in the allotted time. His teammate Jocelyn gives him a task to complete the concept of Postfix to Prefix Conversion for a given expression. can you help him?

Functional Description:

- Read the Prefix expression in reverse order (from right to left)
- If the symbol is an operand, then push it onto the Stack
- If the symbol is an operator, then pop two operands from the Stack
  Create a string by concatenating the two operands and the operator after them.
  string = operand1 + operand2 + operator
  And push the resultant string back to Stack
- Repeat the above steps until end of Prefix expression.

Constraints

the input should be a expressions

Input Format

Single line represents the postfixed expressions

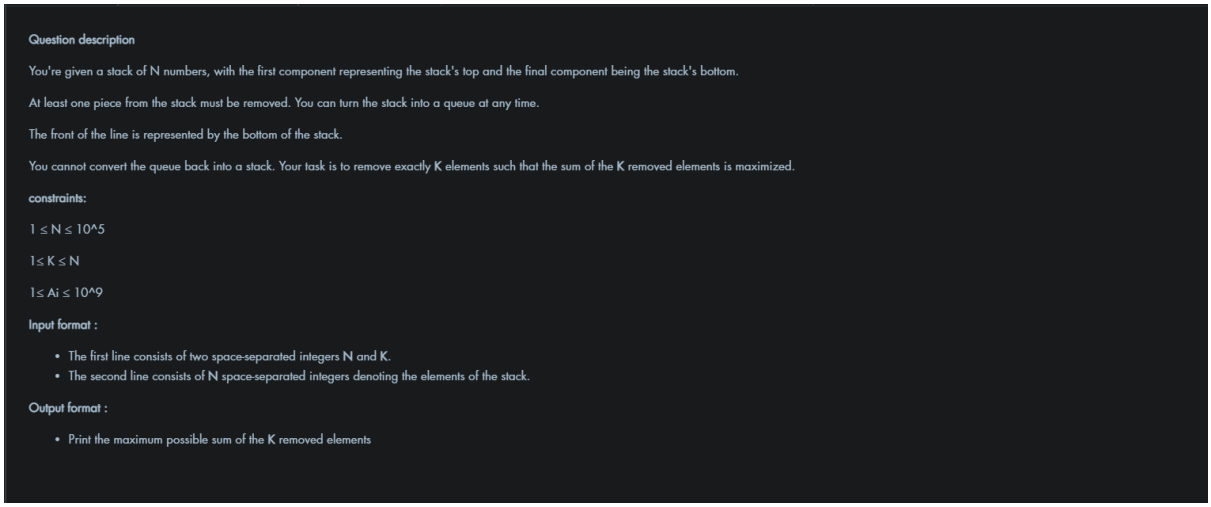
Output Format

Single line represents the prefix expression

```
#include <iostream>
#include <stack>
using namespace std;

bool isOperator(char x)
{
    if (x == '+' || x == '-' || x == '/' || x == '*')
        return true;
    return false;
}

string postToPre(string post_exp)
{
    stack<string> s;

    int length = post_exp.size();

    for (int i = 0; i < length; i++)
    {
        if (isOperator(post_exp[i]))
        {
            string op1 = s.top();
            s.pop();
            string op2 = s.top();
            s.pop();

            string temp = post_exp[i] + op2 + op1;
            s.push(temp);
        }

        else
            s.push(string(1, post_exp[i]));
    }

    string ans = "";
    while (!s.empty())
    {
        ans += s.top();
        s.pop();
    }
    return ans;
}

int main()
{
    string post_exp;
    cin >> post_exp;
    cout << postToPre(post_exp);
    return 0;
}
```


```
#include <iostream>
#include <stack>
using namespace std;

bool isOperator(char x)
{
    if (x == '+' || x == '-' || x == '/' || x == '*') return true;
    return false;
}

string preToInfix(string pre_exp)
{
    stack<string> s;

    int length = pre_exp.size();

    for (int i = length - 1; i >= 0; i--)
    {
        if (isOperator(pre_exp[i]))
        {
            string op1 = s.top();
            s.pop();
            string op2 = s.top();
            s.pop();

            string temp = "(" + op1 + pre_exp[i] + op2 + ")";

            s.push(temp);
        }

        else
        {
            s.push(string(1, pre_exp[i]));
        }
    }
    return s.top();
}

int main()
{
    string pre_exp;
    cin >> pre_exp;
```

```
    cout << "Infix:" << preToInfix(pre_exp);
    return 0;
}
```



Question description

You're given a stack of N numbers, with the first component representing the stack's top and the final component being the stack's bottom.

At least one piece from the stack must be removed. You can turn the stack into a queue at any time.

The front of the line is represented by the bottom of the stack.

You cannot convert the queue back into a stack. Your task is to remove exactly K elements such that the sum of the K removed elements is maximized.

constraints:

1 ≤ N ≤ 10^5

1≤ K ≤ N

1≤ Ai ≤ 10^9

Input format :

- The first line consists of two space-separated integers N and K.
- The second line consists of N space-separated integers denoting the elements of the stack.

Output format :

- Print the maximum possible sum of the K removed elements

```
#include <iostream>
#include <stack>
using namespace std;

int main()
{
    int n, k, i;
    cin >> n >> k;
    int sum = 0;
    int arr[n];
    stack<int> st, st2;
    for (i = 0; i < n; i++)
    {
        cin >> arr[i];
        st.push(arr[i]);
    }
    for (i = 0; i < k; i++)
    {
        st2.push(arr[i]);
        sum += arr[i];
    }
    int maxs = sum;
    while (k-- > 1)
    {
        sum -= st2.top();
        st2.pop();
        sum += st.top();
        st.pop();
        if (sum > maxs)
            maxs = sum;
    }
    cout << maxs;
    return 0;
}
```

Question description

Hassan gets a job in a software company in Hyderabad. The training period for the first three months is 20000 salary. Then incremented to 25000 salaries.

Training is great but they will give you a programming task every day in three months. Hassan must finish it in the allotted time. His teammate Jocelyn gives him a task to complete the concept of Postfix to Infix Conversion for a given expression. can you help him?

Functional Description:

- Read the next symbol from the input.
- Push it onto the stack.
- the symbol is an operator.
- Pop the top 2 values from the stack.
- Put the operator, with the values as arguments and form a string.
- Push the resulted string back to stack.
- If there is only one value in the stack
- That value in the stack is the desired infix string.

Constraints

the input should be a expressions

Input Format

Single line represents the postfixed expressions

Output Format

Single line represents the Infix expression

```
#include <iostream>
#include <stack>
using namespace std;

bool isOperand(char x)
{
    return (x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z');
}

string getInfix(string exp)
{
    stack<string> s;
    for (int i = 0; exp[i] != '\0'; i++)
    {
        if (isOperand(exp[i]))
        {
            string op(1, exp[i]);
            s.push(op);
        }
        else
        {
            string op1 = s.top();
            s.pop();
            string op2 = s.top();
            s.pop();
            s.push("(" + op2 + exp[i] + op1 + ")");
        }
    }
    return (s.top());
}

int main()
{
    string exp;
```

```
    cin >> exp;
    cout << getInfix(exp);
    return 0;
}
```

Question description

Hassan gets a job in a software company in Hyderabad. The training period for the first three months is 20000 salary. Then incremented to 25000 salaries.

Training is great but they will give you a programming task every day in three months. Hassan must finish it in the allotted time. His teammate Jocelyn gives him a task to complete the concept of Prefix to Postfix Conversion for a given expression. can you help him?

Functional Description:

- Read the Prefix expression in reverse order (from right to left)
- If the symbol is an operand, then push it onto the Stack
- If the symbol is an operator, then pop two operands from the Stack
  Create a string by concatenating the two operands and the operator after them.
  string = operand1 + operand2 + operator
  And push the resultant string back to Stack
- Repeat the above steps until end of Prefix expression.

Constraints

the input should be a expressions

Input Format

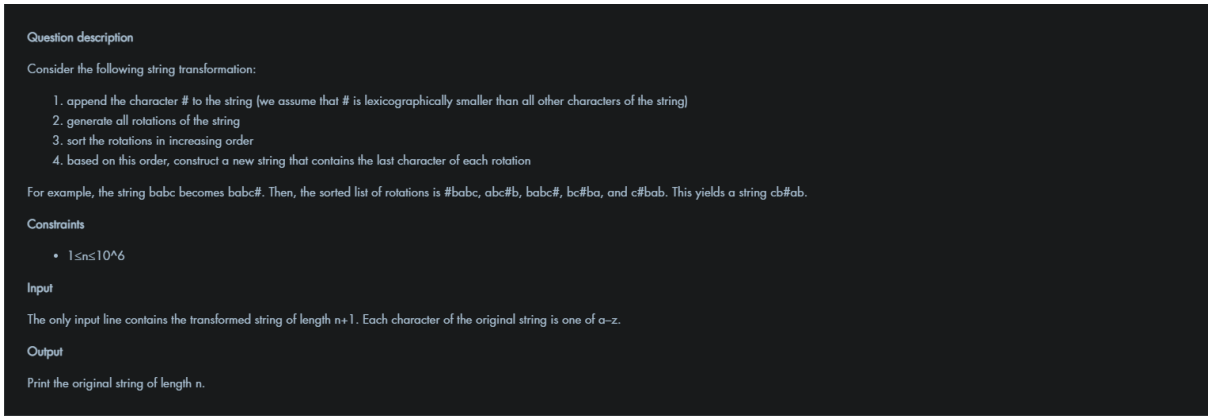
Single line represents the prefixed expressions

Output Format

Single line represents the postfix expression

```
#include <iostream>
#include <stack>
using namespace std;

bool isOperator(char x)
{
    switch (x)
    {
    case '+':
    case '-':
    case '/':
    case '*':
        return true;
    }
    return false;
}

string preToPost(string pre_exp)
{
    stack<string> s;
    int length = pre_exp.size();
    for (int i = length - 1; i >= 0; i--)
    {
        if (isOperator(pre_exp[i]))
        {
            string op1 = s.top();
            s.pop();
            string op2 = s.top();
            s.pop();
            string temp = op1 + op2 + pre_exp[i];
            s.push(temp);
```

```
        }
        else
        {
            s.push(string(1, pre_exp[i]));
        }
    }
    return s.top();
}

int main()
{
    string pre_exp;
    cin >> pre_exp;
    cout << "Postfix:" << preToPost(pre_exp);
    return 0;
}
```



Question description

Consider the following string transformation:

1. append the character # to the string (we assume that # is lexicographically smaller than all other characters of the string)
2. generate all rotations of the string
3. sort the rotations in increasing order
4. based on this order, construct a new string that contains the last character of each rotation

For example, the string babc becomes babc#. Then, the sorted list of rotations is #babc, abc#b, babc#, bc#ba, and c#bab. This yields a string cb#ab.

Constraints

- 1≤n≤10^6

Input

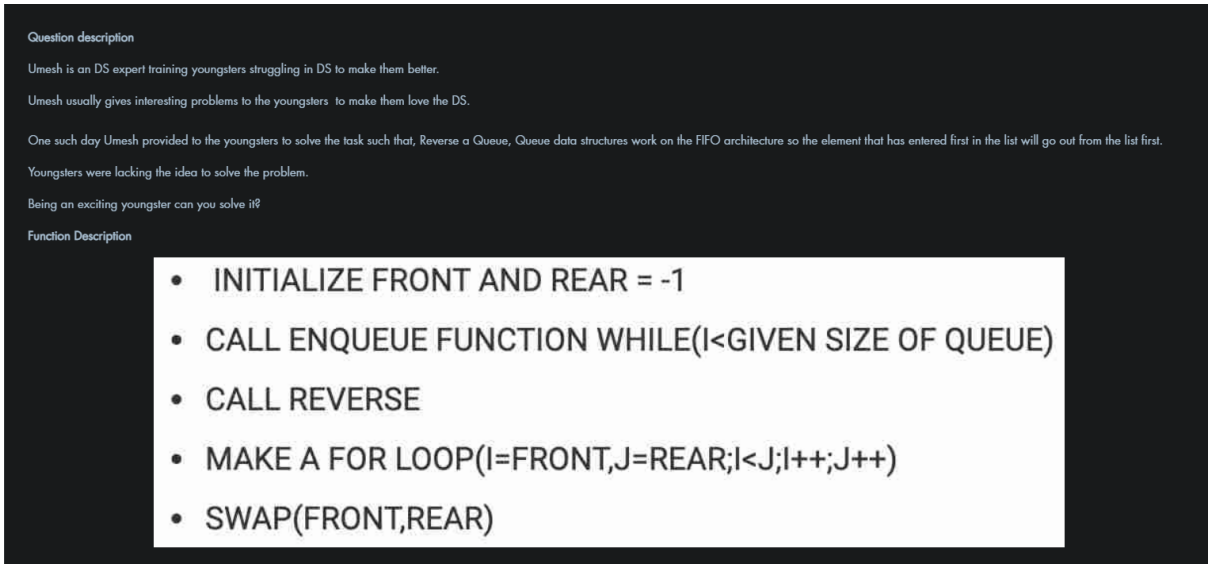
The only input line contains the transformed string of length n+1. Each character of the original string is one of a–z.

Output

Print the original string of length n.

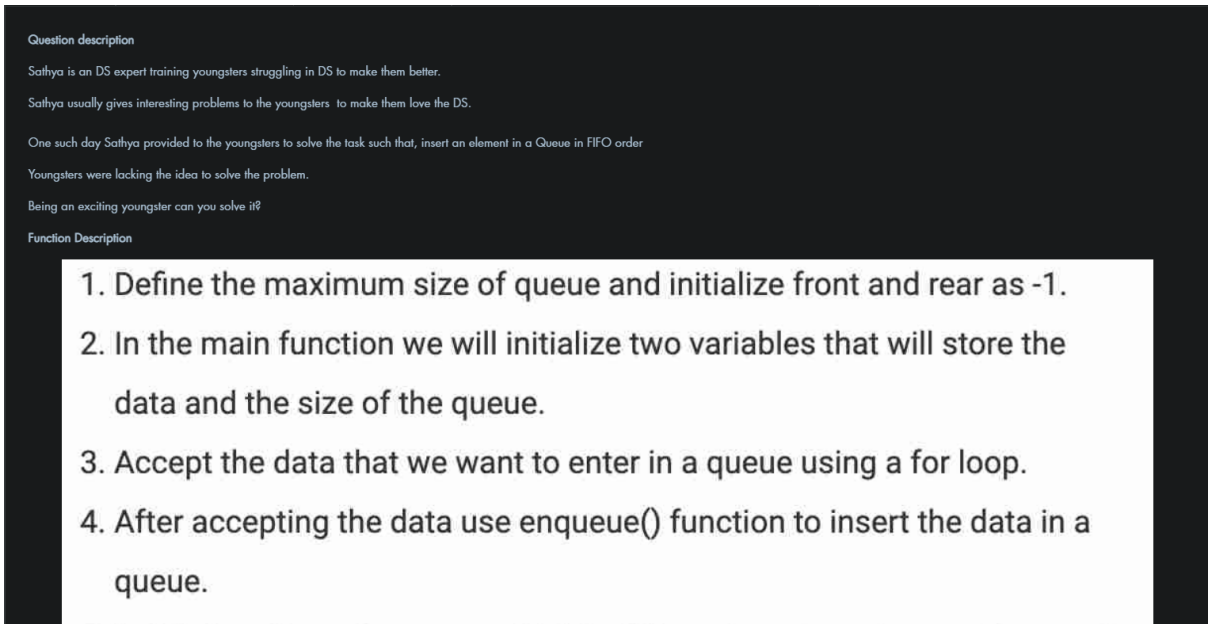
```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    int i;
    string s;
    cin >> s;
    vector<int> v;
    vector<int> a[26];
    int n = s.size();
    for (i = 0; i <= n; i++)
    {
        if (s[i] == '#')
            v.push_back(i);
        else
            a[s[i] - 'a'].push_back(i);
    }
    for (int i = 0; i < 26; i++)
    {
        for (auto j : a[i])
            v.push_back(j);
```

```
    }
    string ans;
    int j = v[v[0]];
    while (s[j] != '#')
    {
        ans += s[j];
        j = v[j];
    }
    cout << ans;
    return 0;
}
```




```
#include <stdio.h>
#define SIZE 100

void enqueue(int data, int l);
void display();
void reverse();
int items[SIZE], front = -1, rear = -1;

int main()
{
    int n, t, i;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &t);
        enqueue(t, n);
    }
    printf("Queue:");
    display();
    reverse();
    printf("\nReversed Queue:");
    display();
    return 0;
}
```

```
void reverse()
{
    int i, j, temp;
    for (i = front, j = rear; i < j; i++, j--)
    {
        temp = items[i];
        items[i] = items[j];
        items[j] = temp;
    }
}

void enqueue(int data, int l)
{
    if (rear == l - 1)
        printf("Queue is Full!");
    else
    {
        if (front == -1)
            front = 0;
        rear++;
        items[rear] = data;
    }
}

void display()
{
    if (rear == -1)
        printf("\nQueue is Empty!");
    else
    {
        int i;
        for (i = front; i <= rear; i++)
            printf("%d ", items[i]);
    }
}
```



```c
#include <stdio.h>
#define SIZE 100
void enqueue(int);
void display();
int items[SIZE], front = -1, rear = -1;

int main()
{
    int n, data, i;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &data);
        enqueue(data);
        display();
    }
    return 0;
}

void enqueue(int data)
{
    if (rear == SIZE - 1)
        printf("Queue is Full!");
    else
    {
        if (front == -1)
            front = 0;
        rear++;
        items[rear] = data;
        printf("Enqueuing %d\n", data);
    }
}

void display()
{
    if (rear == -1)
        printf("\nQueue is Empty!");
    else
    {
        int i;
        for (i = front; i <= rear; i++)
            printf("%d ", items[i]);
    }
}
```

Question description

On the last day of the semester, Shahid's students were talking and playing very loudly to the delight of the end of the semester. The principal of the college severely reprimanded Shahid. But he decided to engage them in a different activity without getting angry at the students.

So Shahid gave his students to solve the task such that, Circular Queue using Linked List and dequeue two elements from circular queue.

there is no memory waste while using Circular Queue, it is preferable than using a regular queue.

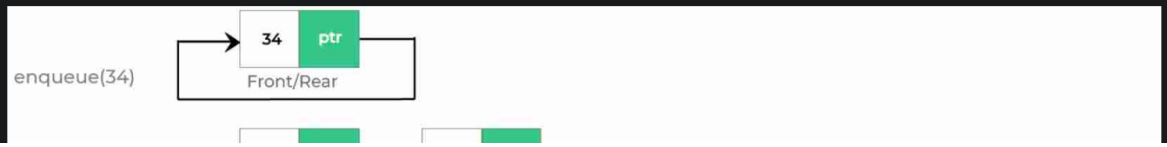
Because linked lists allow for dynamic memory allocation, they are simple to build.

Circular Queue implementation using linked list is identical to circular linked list except that circular Queue has two pointers front and back whereas circular linked list only has one pointer head.

Final Year students were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

Function Description




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

struct node
{
    int data;
    struct node *next;
};
struct node *f = NULL;
struct node *r = NULL;

void linkedListTraversal(struct node *ptr)
{
    while (ptr != NULL)
    {
        printf("%d ", ptr->data);
        ptr = ptr->next;
    }
}

void enqueue(int d)
{
    struct node *n;
    n = (struct node *)malloc(sizeof(struct node));
    if (n == NULL)
    {
        printf("Queue is Full");
    }
    else
    {
        n->data = d;
        n->next = NULL;
        if (f == NULL)
        {
            f = r = n;
        }
        else
        {
            r->next = n;
            r = n;
```

```
        }
    }
}

int dequeue()
{
    int val = -1;
    struct node *t;
    t = f;
    if (f == NULL)
    {
        printf("Queue is Empty\n");
    }
    else
    {
        f = f->next;
        val = t->data;
        free(t);
    }
    return val;
}

int main()
{
    int n, i, t;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &t);
        enqueue(t);
    }
    linkedListTraversal(f);
    for (i = 0; i < 2; i++)
    {
        dequeue();
        printf("\n");
        linkedListTraversal(f);
    }
    return 0;
}
```

Question description

Your task is to construct a tower in N days by following these conditions:

- Every day you are provided with one disk of distinct size.
- The disk with larger sizes should be placed at the bottom of the tower.
- The disk with smaller sizes should be placed at the top of the tower.

The order in which tower must be constructed is as follows:

- You cannot put a new disk on the top of the tower until all the larger disks that are given to you get placed.

Print N lines denoting the disk sizes that can be put on the tower on the ith day.

Constraints:

1≤N≤10^6

1≤size ≤ N

Input format

- First line: N denoting the total number of disks that are given to you in the N subsequent days
- Second line: N integers in which the ith integers denote the size of the disks that are given to you on the ith day

Note: All the disk sizes are distinct integers in the range of 1 to N.

Output format

Print N lines. In the ith line, print the size of disks that can be placed on the top of the tower in descending order of the disk sizes.

If on the ith day no disks can be placed, then leave that line empty.

```
#include <stdio.h>

int main()
{
    long int disk, temp[1000000] = {0}, size, i, max;
    scanf("%ld", &disk);
    max = disk;
    for (i = 0; i < disk; i++)
    {
        scanf("%ld", &size);
        temp[size] = size;
        if (size == max)
        {
            while (temp[size])
            {
                printf("%ld ", temp[size]);
                size--;
            }
            max = size;
            printf("\n");
        }
        else
            printf("\n");
    }
    return 0;
}
```

Question description

Given an array of n integers, your task is to process q queries of the form: what is the minimum value in range [a,b]?

Constraints

- $1 \le n,q \le 2 \cdot 10^5$
- $1 \le x_i \le 10^9$
- $1 \le a \le b \le n$

Input

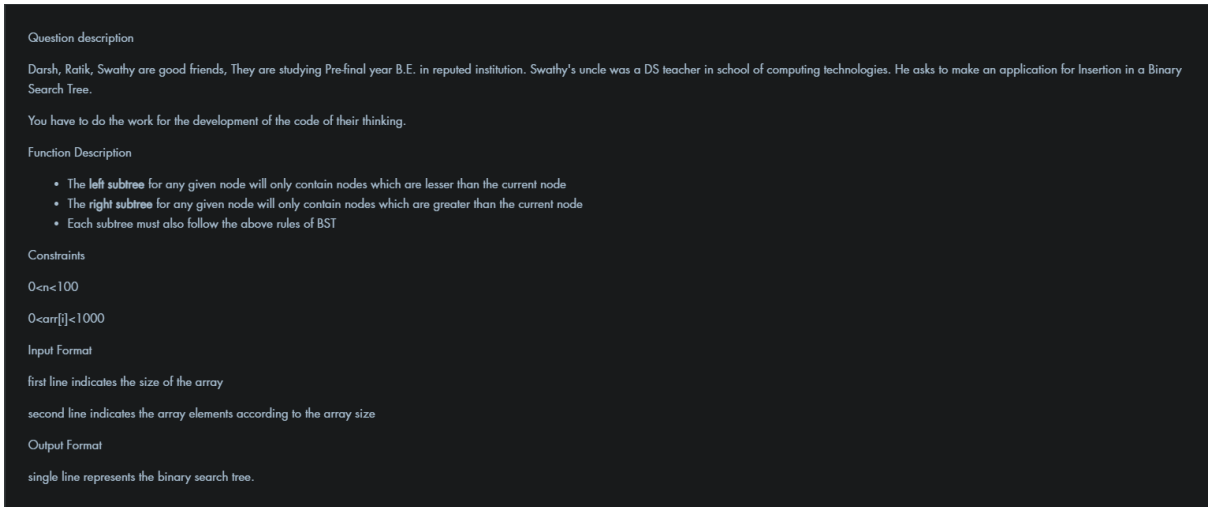
The first input line has two integers n and q: the number of values and queries.
The second line has n integers x1,x2,...,xn: the array values.
Finally, there are q lines describing the queries. Each line has two integers a and b: what is the minimum value in range [a,b]?

Output

Print the result of each query.

```
#include <stdio.h>
#define N 200000
#define N_ (1 << 18)
#define INF 0x3f3f3f3f
int tt[N_ * 2];
void build(int *aa, int k, int l, int r)
{
    int m, k1, k2;

    if (r - l == 1)
    {
        tt[k] = aa[l];
        return;
    }
    m = (l + r) / 2, k1 = k * 2 + 1, k2 = k * 2 + 2;
    build(aa, k1, l, m);
    build(aa, k2, m, r);
    tt[k] = tt[k1] < tt[k2] ? tt[k1] : tt[k2];
}
int query(int k, int l, int r, int ql, int qr)
{
    int m, q1, q2;

    if (qr <= l || r <= ql)
        return INF;
    if (ql <= l && r <= qr)
        return tt[k];
    m = (l + r) / 2;
    q1 = query(k * 2 + 1, l, m, ql, qr);
    q2 = query(k * 2 + 2, m, r, ql, qr);
    return q1 < q2 ? q1 : q2;
}
int main()
{
    static int aa[N];
    int n, q, i, j;

    scanf("%d%d", &n, &q);
    for (i = 0; i < n; i++)
        scanf("%d", &aa[i]);
    build(aa, 0, 0, n);
```

```
    while (q--)
    {
        scanf("%d%d", &i, &j), i--;
        printf("%d\n", query(0, 0, n, i, j));
    }
    return 0;
}
```



Question description

Darsh, Ratik, Swathy are good friends, They are studying Pre-final year B.E. in reputed institution. Swathy's uncle was a DS teacher in school of computing technologies. He asks to make an application for Insertion in a Binary Search Tree.

You have to do the work for the development of the code of their thinking.

Function Description

- The **left subtree** for any given node will only contain nodes which are lesser than the current node
- The **right subtree** for any given node will only contain nodes which are greater than the current node
- Each subtree must also follow the above rules of BST

Constraints

0<n<100

0<arr[i]<1000

Input Format

first line indicates the size of the array

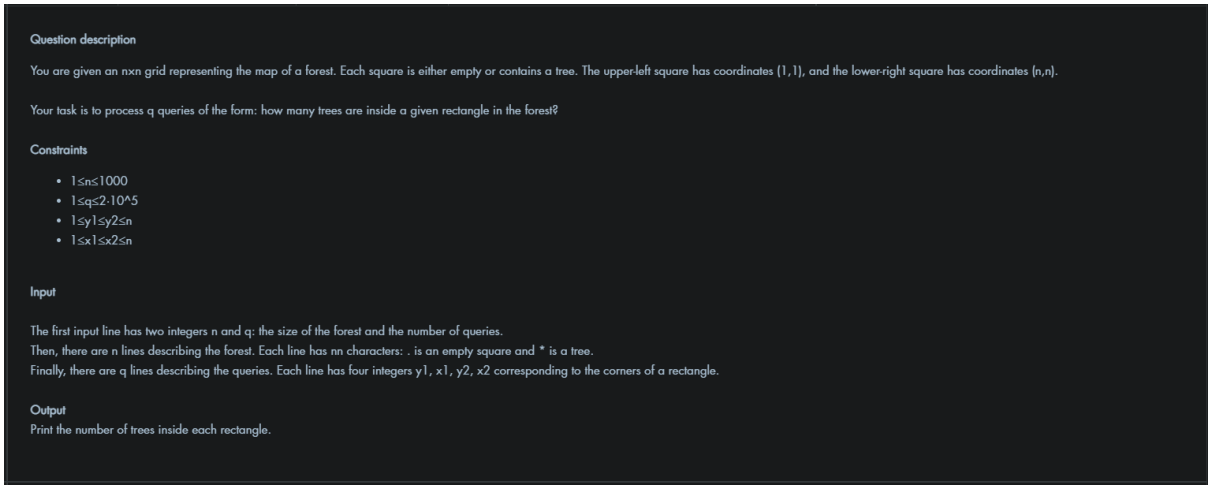
second line indicates the array elements according to the array size

Output Format

single line represents the binary search tree.

```
#include <iostream>
using namespace std;

struct node
{
    int dat;
    struct node *left, *right;
};
struct node *newNode(int item)
{
    struct node *n = new node;
    n->dat = item;
    n->left = NULL;
    n->right = NULL;
    return n;
}
struct node *insertNode(node *head, node *n)
{
    if (head == NULL)
    {
        return n;
    }
    else
    {
        if (head->dat > n->dat)
        {
            head->left = insertNode(head->left, n);
        }
        else
```

```
        {
            head->right = insertNode(head->right, n);
        }

        return head;
    }
}
void dfs(node *head)
{
    if (head == NULL)
        return;

    dfs(head->left);
    cout << head->dat << ' ';
    dfs(head->right);
}
int main()
{
    int n, temp;
    struct node *head = NULL;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> temp;
        head = insertNode(head, newNode(temp));
    }
    dfs(head);

    return 0;
}
```



Question description

You are given an nxn grid representing the map of a forest. Each square is either empty or contains a tree. The upper-left square has coordinates (1,1), and the lower-right square has coordinates (n,n).

Your task is to process q queries of the form: how many trees are inside a given rectangle in the forest?

Constraints

- 1≤n≤1000
- 1≤q≤2·10^5
- 1≤y1≤y2≤n
- 1≤x1≤x2≤n

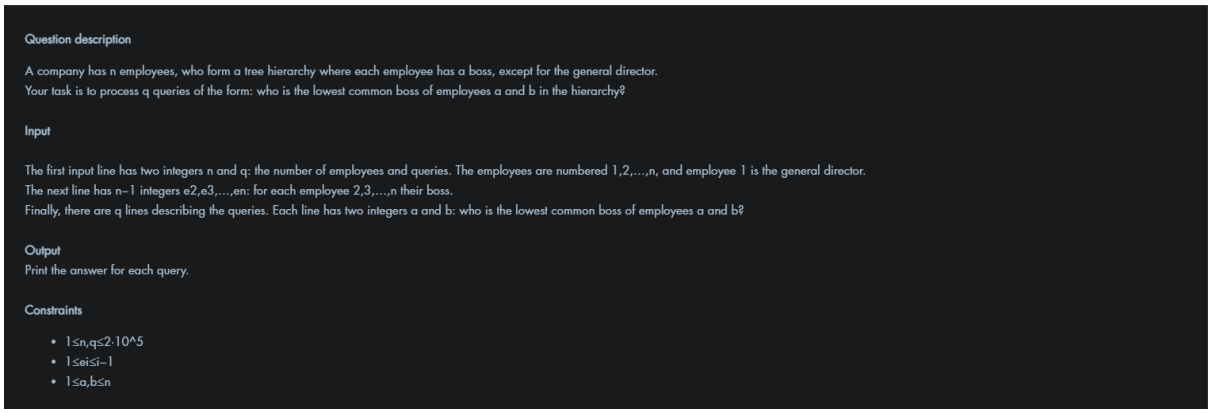
Input

The first input line has two integers n and q: the size of the forest and the number of queries.
Then, there are n lines describing the forest. Each line has nn characters: . is an empty square and * is a tree.
Finally, there are q lines describing the queries. Each line has four integers y1, x1, y2, x2 corresponding to the corners of a rectangle.

Output
Print the number of trees inside each rectangle.

```
#include <iostream>
using namespace std;

int main()
{
    int n, m;
    cin >> n >> m;
```

```
    string s[n];
    int i;
    for (i = 1; i <= n; i++)
        cin >> s[i - 1];

    while (m--)
    {
        int a, b, c, d;
        cin >> a >> b >> c >> d;
        int ans = 0;
        for (int i = a; i <= c; i++)
        {
            for (int j = b; j <= d; j++)
            {
                if (s[i - 1][j - 1] == '*')
                {
                    ans++;
                }
            }
        }
        cout << ans << endl;
    }

    return 0;
}
```



Question description

A company has n employees, who form a tree hierarchy where each employee has a boss, except for the general director.
Your task is to process q queries of the form: who is the lowest common boss of employees a and b in the hierarchy?

Input

The first input line has two integers n and q: the number of employees and queries. The employees are numbered 1,2,...,n, and employee 1 is the general director.
The next line has n−1 integers e2,e3,...,en: for each employee 2,3,...,n their boss.
Finally, there are q lines describing the queries. Each line has two integers a and b: who is the lowest common boss of employees a and b?

Output

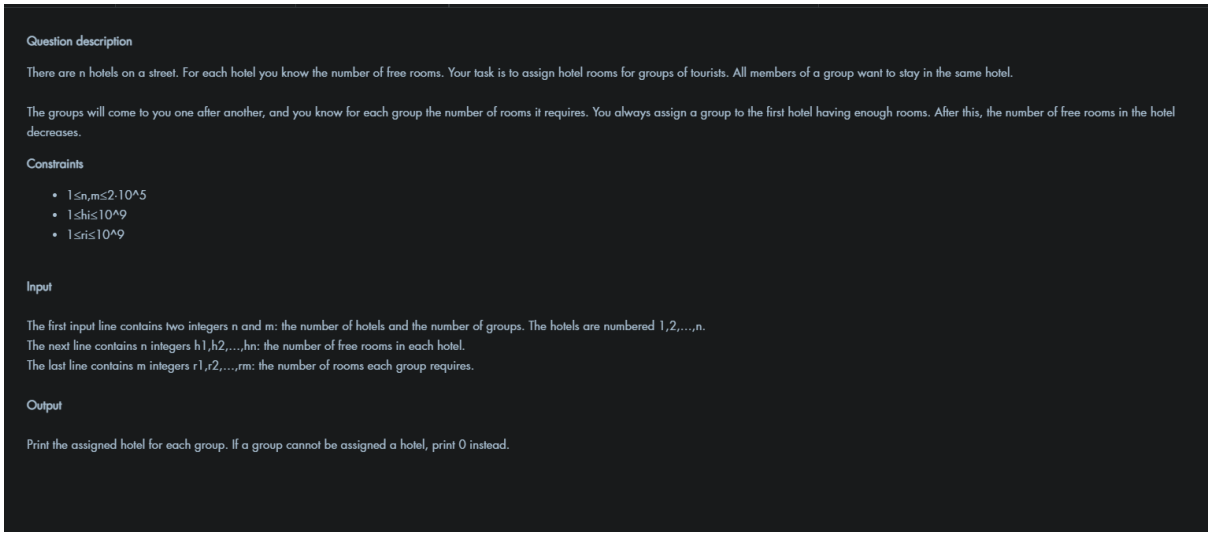
Print the answer for each query.

Constraints

- $1 \le n,q \le 2 \cdot 10^5$
- $1 \le e_i \le i-1$
- $1 \le a,b \le n$

```
#include <stdio.h>
#define N 200000
#define LN 17

int main()
{
    static int dd[N], pp[LN + 1][N];
    int n, q, p, i, j, k, tmp;

    scanf("%d%d", &n, &q);
    for (i = 1; i < n; i++)
    {
        scanf("%d", &p), p--;
        dd[i] = dd[p] + 1;
        pp[0][i] = p;
```

```
    }
    for (k = 1; k <= LN; k++)
        for (i = 0; i < n; i++)
            pp[k][i] = pp[k - 1][pp[k - 1][i]];
    while (q--)
    {
        scanf("%d%d", &i, &j), i--, j--;
        if (dd[i] < dd[j])
            tmp = i, i = j, j = tmp;
        if (dd[i] != dd[j])
            for (k = LN; k >= 0; k--)
                if (1 << k <= dd[i] - dd[j])
                    i = pp[k][i];
        if (i != j)
        {
            for (k = LN; k >= 0; k--)
                if (1 << k <= dd[i] && pp[k][i] != pp[k][j])
                {
                    i = pp[k][i];
                    j = pp[k][j];
                }
            i = pp[0][i];
        }
        printf("%d\n", i + 1);
    }
    return 0;
}
```



Question description

There are n hotels on a street. For each hotel you know the number of free rooms. Your task is to assign hotel rooms for groups of tourists. All members of a group want to stay in the same hotel.

The groups will come to you one after another, and you know for each group the number of rooms it requires. You always assign a group to the first hotel having enough rooms. After this, the number of free rooms in the hotel decreases.

Constraints

- $1 \le n,m \le 2 \cdot 10^5$
- $1 \le h_i \le 10^9$
- $1 \le r_i \le 10^9$

Input

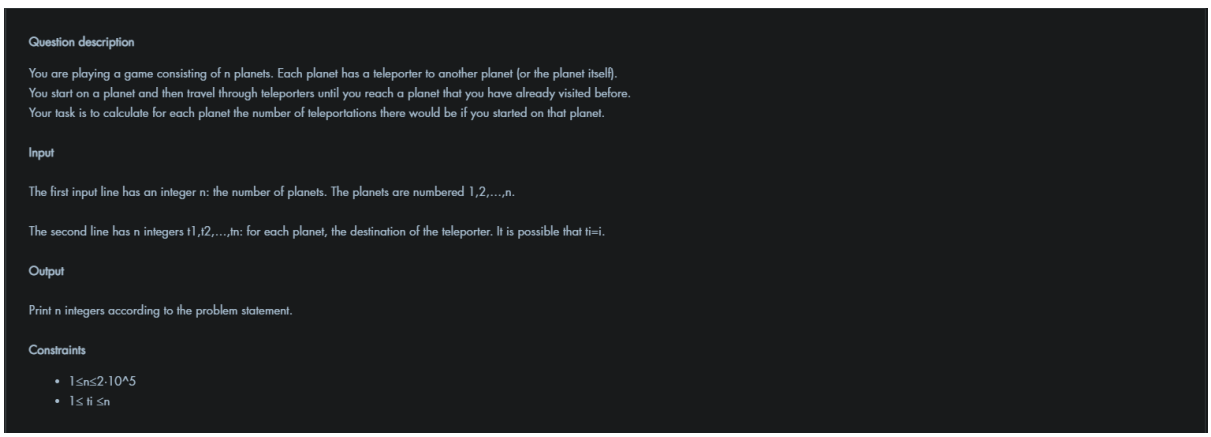
The first input line contains two integers n and m: the number of hotels and the number of groups. The hotels are numbered 1,2,...,n.
The next line contains n integers h1,h2,...,hn: the number of free rooms in each hotel.
The last line contains m integers r1,r2,...,rm: the number of rooms each group requires.

Output

Print the assigned hotel for each group. If a group cannot be assigned a hotel, print 0 instead.

```
#include <stdio.h>
#define N 200000
#define N_ (1 << 18)
int tr[N_ * 2], hh[N];
void build(int k, int l, int r)
{
    if (r - l == 1)
        tr[k] = hh[l];
    else
```

```
    {
        int m = (l + r) / 2, k1 = k * 2 + 1, k2 = k * 2 + 2;

        build(k1, l, m);
        build(k2, m, r);
        tr[k] = tr[k1] > tr[k2] ? tr[k1] : tr[k2];
    }
}
int update(int k, int l, int r, int x)
{
    int m, k1, k2, i;

    if (r - l == 1)
    {
        tr[k] -= x;
        return l;
    }
    m = (l + r) / 2, k1 = k * 2 + 1, k2 = k * 2 + 2;
    i = tr[k1] >= x ? update(k1, l, m, x) : update(k2, m, r, x);
    tr[k] = tr[k1] > tr[k2] ? tr[k1] : tr[k2];
    return i;
}
int main()
{
    int n, m, i;

    scanf("%d%d", &n, &m);
    for (i = 0; i < n; i++)
        scanf("%d", &hh[i]);
    build(0, 0, n);
    while (m--)
    {
        int x, i;

        scanf("%d", &x);
        i = tr[0] >= x ? update(0, 0, n, x) + 1 : 0;
        printf("%d ", i);
    }
    printf("\n");
    return 0;
}
```



Question description

You are playing a game consisting of n planets. Each planet has a teleporter to another planet (or the planet itself).
You start on a planet and then travel through teleporters until you reach a planet that you have already visited before.
Your task is to calculate for each planet the number of teleportations there would be if you started on that planet.

Input

The first input line has an integer n: the number of planets. The planets are numbered 1,2,...,n.

The second line has n integers t1,t2,...,tn: for each planet, the destination of the teleporter. It is possible that ti=i.

Output

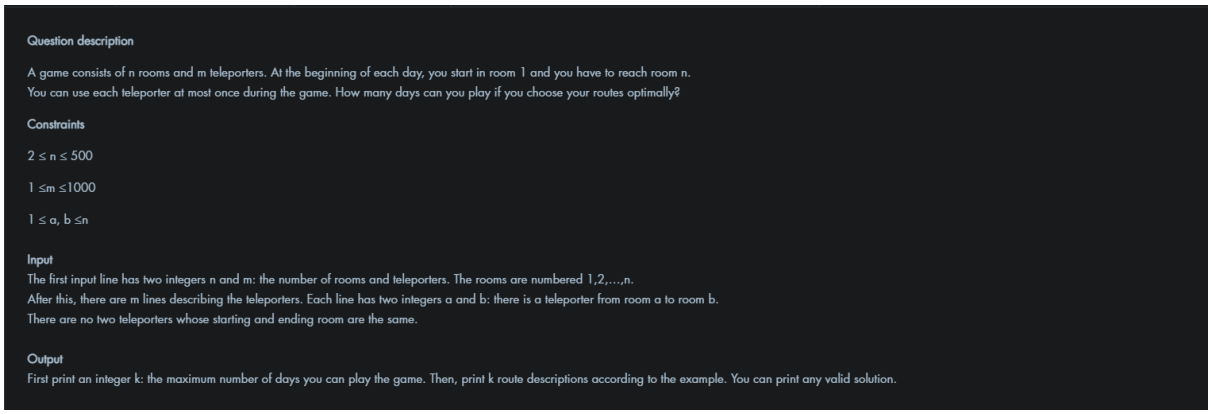
Print n integers according to the problem statement.

Constraints

- 1≤n≤2·10^5
- 1≤ ti ≤n

```
#include <stdio.h>
#include <string.h>
#define N 200000

int main()
{
    static int aa[N], cc[N], dd[N], qq[N];
    int n, i, j, c, d, q, cnt;

    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &aa[i]), aa[i]--;
    memset(cc, -1, n * sizeof *cc);
    cnt = 0;
    for (i = 0; i < n; i++)
    {
        if (cc[i] != -1)
            continue;
        d = 0;
        j = i;
        while (cc[j] == -1)
        {
            cc[j] = -2;
            d++;
            j = aa[j];
        }
        if (cc[j] == -2)
        {
            c = cnt++;
            q = 0;
            while (cc[j] == -2)
            {
                cc[j] = c;
                q++;
                j = aa[j];
            }
            qq[c] = q;
            d -= q;
        }
        else
        {
            c = cc[j];
            d += dd[j];
        }
        j = i;
        while (cc[j] == -2)
        {
            cc[j] = c;
            dd[j] = d--;
            j = aa[j];
        }
    }
    for (i = 0; i < n; i++)
        printf("%d ", dd[i] + qq[cc[i]]);
    printf("\n");
```

```
    return 0;
}
```



Question description

A game consists of n rooms and m teleporters. At the beginning of each day, you start in room 1 and you have to reach room n.
You can use each teleporter at most once during the game. How many days can you play if you choose your routes optimally?

Constraints

$2 \le n \le 500$

$1 \le m \le 1000$

$1 \le a, b \le n$

Input

The first input line has two integers n and m: the number of rooms and teleporters. The rooms are numbered 1,2,...,n.
After this, there are m lines describing the teleporters. Each line has two integers a and b: there is a teleporter from room a to room b.
There are no two teleporters whose starting and ending room are the same.

Output

First print an integer k: the maximum number of days you can play the game. Then, print k route descriptions according to the example. You can print any valid solution.

```
#include <stdio.h>
#define N 500
#define M 1000

struct L
{
    struct L *next;
    int h;
} aa[N * 2];

int ij[M + N], cc[(M + N) * 2], dd[N * 2];

int bfs(int n, int s, int t)
{
    static int qq[N * 2];
    int head, cnt, h, i, j, d;

    for (i = 0; i < n; i++)
        dd[i] = n;
    dd[s] = 0;
    head = cnt = 0;
    qq[head + cnt++] = s;
    while (cnt)
    {
        struct L *l;

        i = qq[cnt--, head++];
        d = dd[i] + 1;
        for (l = aa[i].next; l; l = l->next)
            if (cc[h = l->h])
            {
                j = i ^ ij[h >> 1];
                if (dd[j] == n)
                {
                    dd[j] = d;
                    if (j == t)
                        return 1;
```

```
                    qq[head + cnt++] = j;
                }
            }
    }
    return 0;
}

int dfs(int n, int i, int t)
{
    struct L *l;
    int h, j, d;

    if (i == t)
        return 1;
    d = dd[i] + 1;
    for (l = aa[i].next; l; l = l->next)
        if (cc[h = l->h])
        {
            j = i ^ ij[h >> 1];
            if (dd[j] == d && dfs(n, j, t))
            {
                cc[h]--, cc[h ^ 1]++;
                return 1;
            }
        }
    dd[i] = n;
    return 0;
}

int dinic(int n, int s, int t)
{
    int f = 0;

    while (bfs(n, s, t))
        while (dfs(n, s, t))
            f++;
    return f;
}

void link(int i, int j, int h, int c)
{
    static struct L l91[(M + N) * 2], *l = l91;

    ij[h] = i ^ j;
    cc[h << 1] = c;
    l->h = h << 1;
    l->next = aa[i].next, aa[i].next = l++;
    l->h = h << 1 ^ 1;
    l->next = aa[j].next, aa[j].next = l++;
}
int qq[N];

int path(int i, int t)
{
    int cnt = 0;

    while (i != t)
    {
```

```
        struct L *l;
        int h;

        qq[cnt++] = i;
        for (l = aa[i].next; l; l = l->next)
            if (((h = l->h) & 1) == 0 && cc[h ^ 1])
            {
                cc[h]++, cc[h ^ 1]--;
                i ^= ij[h >> 1];
                break;
            }
    }
    qq[cnt++] = t;
    return cnt;
}

int main()
{
    int n, m, h, i, j, k, s, t, cnt;

    scanf("%d%d", &n, &m);
    for (h = 0; h < m; h++)
    {
        scanf("%d%d", &i, &j), i--, j--;
        link(i << 1 ^ 1, j << 1, h, 1);
    }
    for (i = 0; i < n; i++)
        link(i << 1, i << 1 ^ 1, m + i, n);
    s = 0, t = (n - 1) << 1 ^ 1;
    k = dinic(n * 2, s, t);
    printf("%d\n", k);
    while (k--)
    {
        cnt = path(s, t);
        printf("%d\n", cnt / 2);
        for (i = 0; i < cnt; i += 2)
            printf("%d ", (qq[i] >> 1) + 1);
        printf("\n");
    }
    return 0;
}
```

Problem Description

Little Chandan is an exceptional manager - apart from his role in university as the person who has to bug everyone, in general... and if possible, try to get some work done.

He's also offered a job as the coach of the best Russian teams participating for ACM-ICPC World Finals. Now, Chandan is an extremely good coach, too. But he's a weird person who thrives on patterns in life, in general. So, he has decided that if there are *n* number of students in total, and he is supposed to divide them in camps of *k* students - he want them to be arranged in such a way that the length of names of all the students in a camp is **equal.**

I know, totally weird, right?

Constraints:
1 <= **Test Cases** <= 50
1 <= **N** <= 1000
1 <= **K** <= 1000
1 <= **LengthOfAString** <= 100
The name of a programmer will always be in lower case.

Input:
The first line will contain the number of test cases. Which will be followed by two integers, *n*, *k* - denoting the number of total students, and the number of total students which will be allowed in one camp. After which, n lines will follow denoting the names of all the students who're willing to learn by the great coach.

Output:
If it is possible for *all* the students be arranged in a camp of *k* students, print "Possible", else print "Not possible".

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

int main()
{
    int cases, N, K, i, j, len, bins[100], flag;
    scanf("%d", &cases);
    int results[cases];

    for (i = 0; i < cases; i++)
    {
        flag = 0;
        for (j = 0; j < 100; j++)
            bins[j] = 0;

        scanf("%d %d", &N, &K);
        char str[N][100];

        for (j = 0; j < N; j++)
        {
            scanf("%s", str[j]);
            len = strlen(str[j]);
            bins[len] += 1;
        }

        for (j = 0; j < 100; j++)
        {
            if (bins[j] % K != 0)
            {
                results[i] = 0;
                flag = 1;
                break;
            }
        }
        if (flag == 0)
            results[i] = 1;
    }

    for (i = 0; i < cases; i++)
```

```
    {
        if (results[i] == 0)
            printf("Not possible\n");
        else
            printf("Possible\n");
    }
    return 0;
}
```

Problem Description:

Rick was besieged by walkers after the Governor's raid on the prison. They are approaching him from all sides. Assume Rick possesses a limitless supply of ammunition. Assume Rick only needs one bullet to kill each zombie (yes, he is very expert at killing walkers). They must be shot in the head. Take a look at how excellent he is).

As soon as he kills one walker, the remainder of the zombies advance 1 metre. There are n walkers in front of Rick, each at a different distance. Rick will perish if any walker is able to reach him. You must now determine whether he will live or die. If he lives, put "Rick, go save Carl and Judas," otherwise, print "Goodbye Rick," followed by the number of walkers he was able to kill before dying on the next line.

Rick's gun can also fire 6 rounds without reloading. He reloads in 1 second, during which time walkers advance 1 metre.

Constraints
1<=t<=100
1<=n<=100000
1<=dis[i]<=50000

Input Format
First line contains an integer t indicating number of test cases.

Next line contains an integer n denoting no.of walkers followed by n space separated integers denoting the distance of walkers from him.

Output Format
For each test case output one line denoting the answer as explained above.

```
#include <iostream>
#include <string.h>
using namespace std;

void solve() {}

int main()
{
    solve();
    int T;
    cin >> T;
    while (T--)
    {
        bool ans = true;
        int val = 0;
        int n;
        cin >> n;
        int temp;
        int mx[50001], cnt[50001];
        memset(mx, 0, sizeof(mx));
        memset(cnt, 0, sizeof(cnt));
        int tp = 2;
        mx[0] = 1;
        for (int i = 1; i < 50001; i++)
        {
            mx[i] = tp;
            if (tp % 6 == 0)
            {
                i++;
```

```
                mx[i] = tp;
            }
            tp++;
        }
        for (int i = 0; i < n; i++)
        {
            cin >> temp;
            temp--;
            cnt[temp]++;
        }
        for (int i = 0; i < 50001; i++)
        {
            if (i > 0)
                cnt[i] += cnt[i - 1];
            if (cnt[i] > mx[i])
            {
                ans = false;
                val = i;
                break;
            }
        }
        if (ans)
            cout << "Rick now go and save Carl and Judas" << endl;
        else
        {
            val = mx[val];
            cout << "Goodbye Rick\n"
                 << val << endl;
        }
    }
    return 0;
```