Question Description:
Simon has given N ratios in the form of A and B that is represented as A/B. The values of A and B are represented as double data type values. The values of B are incorrect. The actual values of B are B+R. Simon know the actual sum of all the ratios that is available in variable K.

Note: The true values of B, represented as (B+R), are always greater than 0. Simon's task is to determine the value of R.

Constraints:
1 <= N <= 1000
1 <= A <= 1000
|B| <= 1000
1 <= K <= 10^6

Input Format:
First line: Two integers N and col denoting the number of ratios and the value 2 respectively
Next N lines: Each line contains two double values A and B
Last line: A double value K denoting the sum of all the ratios

Output Format:
Print the value of R. Simon's answer must contain an absolute or relative error of less than 10^-6.

```
#include<iostream>


using namespace std;


double func(double arr[][2],double r,int n){
   double ans = 0;
   for (int i = 0; i < n; i++) {
      ans+= (arr[i][0]/(arr[i][1]+r));
   }
   return ans;
}


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

```
    double arr[n][2];
    for (int i = 0; i < n; i++) {
        cin>>arr[i][0]>>arr[i][1];
    }
    double hi=2000,lo=0,mid,curr,k;
    cin>>k;
    while(hi-lo>1e-7){
        mid=(hi+lo)/2;
        curr=func(arr,mid,n);
        if(curr<k){
            hi = mid;
        }
        else{
            lo = mid + 1e-7;
        }
    }
    printf("%.6f",mid);


    return 0;
    printf("double solve(double** arr,double K,int n)");
}
```

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:
$1 \leq T \leq 100$.
$1 \leq Xi \leq D$.

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 <iostream>

using namespace std;

int main()

{

   int T;

   cin>>T;

   for(int t=0;t<T;t++){

      int n,d;

      cin>>n>>d;

      int x[n];

      for(int i=0;i<n;i++){

         cin>>x[i];

      }

      for(int i=n-1;i>=0;i--){

         int temp=(d-(d%x[i]));

         d=temp;


      }

      cout<<d<<endl;



   }

   return 0;
```

}

VIBGYOR isn't just an acronym, it's a way of life for Asian paint company. The owner is considering modernizing his paint mixing equipment with a computerized model. He's hired you to code the prototype. Your simple program will need to correctly output the right color based on the blends he's given you.

Example Colors

Primary colors " RED, BLUE, YELLOW",

secondary Colors "ORANGE, PURPLE, GREEN"

Tertiary Colors " LIGHT RED, DARK RED, LIGHT PURPLE, DARK PURPLE, LIGHT BLUE, DARK BLUE, LIGHT GREEN, DARK GREEN, LIGHT YELLOW, DARK YELLOW, LIGHT ORANGE, DARK ORANGE"

Input Format:
You will receive one to five lines of color combinations consisting of primary colors and secondary colors as well as black and white to make "dark" and "light" colors. The full science of colorisation and pigments will be implemented next, if your prototype is successful.

Output Format:
Print the output in a separate lines contains, Your program should output the correct color depending on what two colors were "mixed" on the line. Primary colors should mix together to create secondary colors. Anything mixed with "WHITE" or "BLACK" should be output as either "LIGHT X" or "DARK X" where X is the color "WHITE" or "BLACK" were mixed with. Anything mixed with itself won't change colors. You are guaranteed not to receive incompatible colors, or colors not listed in the color wheels shown above (aside from "WHITE" and "BLACK").

Refer logical test cases for your reference.

```
#include <stdio.h>

#include<bits/stdc++.h>

using namespace std;

void arr()

{

   return;

}

int main()
```

```
{
      string ss[] = {"RED", "BLUE", "PURPLE", "YELLOW", "ORANGE" "GREEN"};
         string s,s1;
         int t = 4;
         while(t--)
         {
              cin>>s>>s1;
              //cout<<s<<" "<<s1;
            if(s == ss[0] && s1 == ss[3])
               cout<<"ORANGE";
               else if(s == ss[1] && s1 == ss[3]) cout<<"GREEN";
               else if(s == ss[1] && s1== ss[0]) cout<<"PURPLE";
               else if(s == "BLACK") cout<<"DARK"<<" "<<s1;
                else if(s1 == "BLACK") cout<<"DARK"<<" "<<s;
                else if(s1 == "WHITE") cout<<"LIGHT"<<" "<<s;
                 else if(s == "WHITE") cout<<"LIGHT"<<" "<<s1;
                 else if(s1 == s)cout<<s;
                 else cout<<"N/A";
                  cout<<"\n";
            }
  return 0;
  cout<<"if(strcmp(c,colors[i])==0) for(i=0;i<8;i++) char mixes[8][8][32] char colors[8][32]";}
```

Trapped by a lake and racing against time, our fearless heroes need to quickly cross it in order to stop father from placing the wrong burger order. (Beautiful story, turns out Mike was only joking about the shark).

Unexpected, our heroes have found a ramp on their side of the lake (what could go wrong?). Help them figure out if they can jump the lake (stunts performed on closed course by Peter Hein).

Constraints:
Name = a to z & A to Z
1 <= length <= 500
0 <= rate <= 10 (including decimal)
0 <= width <= 500 (including decimal)

Functional Constraints

if(distance<(width-5.0)) print "SPLASH!"

if((distance>=(width-5.0))&&(distance<=width)) print "JOVA MADE IT!"

if(distance>width) print "LIKE A LEGEND!"

Input Format:
First line of the input is a name of the vehicle
Second line of the input is a length of the ramp (in meters, always a whole 32-bit integer)
Third line of the input is a acceleration rate of the vehicle (in meters/second-squared, floating point decimal of max size
2147483647.0)
Third line of the input is a width of the lake (in meters, floating point decimal of max size 2147483647.0)

Output Format:
Print the output in a single line contains calculate the horizontal speed (rounded to the nearest hundredth) the vehicle will be going
when it runs out of ramp, and then use that to calculate how much horizontal distance (rounded to the nearest tenth) your vehicle will be able to cover (formulas in the discussion section) and output the results of your ramp jumping!

```
#include <stdio.h>

#include<math.h>

int main()

{

   char s[100];

   scanf("%s",s);

   int len;

   float acc,dist,speed,ansdist;

   scanf("%d %f %f",&len,&acc,&dist);

   speed = sqrt(2.0*acc*len);ansdist = speed*speed/9.805;

   printf("%s will reach a speed of %.2f m/s on a %d ramp crossing %.1f of %.1f meters,
",s,speed,len,ansdist,dist);

   if(ansdist<(dist-5.0))

      printf("SPLASH!");

   else if(ansdist>=(dist-5.0)&&ansdist<=dist)

      printf("JOVA MADE IT!");
```

```
   else
      printf("LIKE A LEGEND!");
          return 0;printf("distance=speed1*speed1/9.805;");
}
```

Dr. Ramesh is a professor at a university. He is eager to put on a show for pupils as well. During his lunch break, he decided to host a mind-body activity.

He needs to ask a few thought-provoking questions.

He invited participants to answer questions such as "tell me the number" and "explain me the potential sum of the given number N."

Example Input:

125

Sample output:

8 9 10 11 12 13 14 15 16 17

23 24 25 26 27

62 63

Constraints:

1<N<1000

Input Format:

Single line integer get from user

Output Format:

Display the possible sum of numbers equal to given numbers.

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

void printSums(int N)
{
   int start=1, end=(N+1)/2;
   while (start<end)
```

```
    {
        int sum=0;
        for (int i=start;i<=end;i++)
        {
            sum=sum+i;
            if (sum == N)
            {
                for (int j=start;j<=i;j++)
                    cout <<j<<" ";

                cout <<"\n";
                break;
            }
            if (sum>N)
                break;
        }
        sum=0;
        start++;
    }
}
int main()
{
    int n;
    cin>>n;
    printSums(n);
    return 0;}
```

Tina has been given an array of numbers "A," and she must discover the largest sum that can be attained by selecting a non-empty subset of the array. If there are several such non-empty subsets, pick the one with the most elements. In the specified subset, print the maximum sum and the number of entries.

Constraints:
$1 \leq N \leq 10^5$

-10^9 ≤ Ai ≤ 10^9

Input Format:
The first line contains an integer 'N', denoting the number of elements of the array. Next line contains 'N' space-separated integers, denoting the elements of the array.

Output Format:
Print two space-separated integers, the maximum sum that can be obtained by choosing some subset and the maximum number of elements among all such subsets which have the same maximum sum.

```
#include <stdio.h>

int main()

{

   int cnt=0,temp,tot=0,n;

   scanf("%d",&n);

   while(n--){

      scanf("%d",&temp);

      if(temp>=0){

         cnt++;

         tot+=temp;

      }

   }

   printf("%d %d",tot,cnt);

           return 0;

           printf("if(cnt==0) while(num) ");

}
```

Ragu has given a range [L, R] to Smith. Smith wants to require to find the number of integers 'X' in the range such that GCD(X, F(X)) > 1 where F(x) is equal to the sum of digits of 'X' in its hexadecimal (or base 16) representation.

Example : F(27) = 1+B=1+11=12

(27 in hexadecimal is written as 1B)

Constraints:

1 <= T <= 50

1 <= L

R <= 10^5

Input Format:

The first line contains a positive integer 'T' denoting the number of questions that you are asked.

Each of the next 'T' lines contain two integers 'L' and 'R' denoting the range of questions.

Output Format:

Print the output in a separate lines exactly 'T' numbers as the output.

Sample Input:

3

1 3

5 8

7 12

Sample output:

2

4

6

Explanation

For the first test-case, numbers which satisfy the criteria specified are : 2,3. Hence, the answer is 2.

For the second test-case, numbers which satisfy the criteria specified are : 5,6,7 and 8. Hence, the answer is 4.

For the third test-case, numbers which satisfy the criteria specified are : 7,8,9,10,11 and 12. Hence, the answer is 6.

```
#include <bits/stdc++.h>

using namespace std;

int hexa(int a)

{
int x = 16, ans = 0, rem = 0;
while (a > 0)

{
rem =a % x;
ans += rem;
a/=x;

}

return ans;
}int search(int a, int b)
{ if(__gcd(a, b)>1) return 1; else return 0;}
int main(){
int t;
cin >> t;
while (t--) {
int a, b, count = 0;
cin >> a >> b;
for (int i = a; i <= b; i++){
if (search(i, hexa(i)))
count++; }
cout << count << endl; }
```

```
return 0;

}
```

King Alexander wants every chariot to line up for the start of her Winter-eve ball.

He has asked you, Twilight Sparkle, to sort the horse chariots alphabetically but with royalty in front. Royal horses chariot have diamonds in their names.

Constraints:
1 <= Names <= 100

Input Format:
list of horse chariots numbering anywhere from 1 to 40 horse chariot, with 1 per line. The end of input will be marked with END on a line by itself. Names should be no longer than 100 characters in length, and to only contain letters and spaces.

Output Format:
Print the output in a separate lines contains Sort and list the horse chariots alphabetically in ascending order ('A' "first"), ignoring case. However, any horse chariot with a diamond in their name must be placed at the "top" of the list (before the "A's" start) in the diamond order given in the Discussion section below.

Explanation:

The Pony ranking (in Ascending order) for gemstones is a follows:

Lapis, Topaz, Tourmaline, Sapphire, Peridot, Ruby, Pearl, Emerald, Diamond, Aquamarine, Amethyst, Garnet.

you are guaranteed that you will not have to deal with any gemstones not listed above. If multiple gems are listed in the same name, sort by whichever gem has "highest prioity" (e.g. a pony named 'Garnet Lapis' would be listed before a pony named 'Topaz Sapphire,' because Lapis has the highest priority).

In the case of equal ranking on gemstones, output in alpha order by total name (not just the gemstones), only sort by highest priority gemstones, after that by alpha. (Example, given the names: Lapis Topaz and Amethyst Lapis, they should be printed in order as: Amethyst Lapis then Lapis Topaz. That is because both names have the highest ranked gemstone (Lapis), so we simply sort them in ascending order alphabetically after their ranking has been established. We would not list Lapis Topaz before Amethyst Lapis because Topaz has a higher ranking over Amethyst. Stop comparing gemstone ranking after determine the highest rank of the gems in the name.)

You are guaranteed that there will be no ties in priority for gemstone ranking. You are also guaranteed that you will not encounter hyphenated names like Ruby-Sue. However, if a gemstone name happened to be found as part of a name (like Rubyanne) you can safely treat that as just another name to alphabetise. Gemstone names have to stand on their own (separated by spaces, or the entirety of the name) to be treated royally.

```
#include <bits/stdc++.h>

using namespace std;

#pragma GCC diagnostic ignored "-Wwrite-strings"


//char
*gems[]={"NONE","Garnet","Amethyst","Aquamarine","Diamond","Emerald","Pearl","Ruby","Perido
t","Sapphire","Tourmaline","Topaz","Lapis",0};

string
gems[]={"Garnet","Amethyst","Aquamarine","Diamond","Emerald","Pearl","Ruby","Peridot","Sapphi
re","Tourmaline","Topaz","Lapis"};

int index(string s){

   for (int i = 11; i>0; i--) {

      if(s.find(" "+gems[i]) != string::npos||s.find(gems[i]+" ") != string::npos){

         return 11-i;

      }

   }

   return 12;

}

int main()

{

   vector<string> arr[13];

   string temp;

   while(1){

      getline(cin,temp);

      if(temp=="END") break;

      arr[index(temp)].push_back(temp);

   }

   for (int i = 0; i < 13; i++) {

      sort(arr[i].begin(),arr[i].end());

      for(auto s:arr[i]){

         cout<<s<<endl;

      }
```

```
    }


        return 0;

        printf("char ponies[MAXP][BUFLEN];strcmp(ponies[a],ponies[b])>0;");

        printf("THIS IS THE PROBLEM char
*gems[]={\"NONE\",\"Garnet\",\"Amethyst\",\"Aquamarine\",\"Diamond\",\"Emerald\",\"Pearl\",\"
Ruby\",\"Peridot\",\"Sapphire\",\"Tourmaline\",\"Topaz\",\"Lapis\",0};\"");

    char
*gems[]={"NONE","Garnet","Amethyst","Aquamarine","Diamond","Emerald","Pearl","Ruby","Perido
t","Sapphire","Tourmaline","Topaz","Lapis",0};

    char x = gems[0][0];

    printf("%c",x);

}
```

Siva has several containers, each with a number of fruits in it. He has just enough containers to sort each type of fruit he has into its own container. Siva wants to sort the fruits using his sort method.

Siva wants to perform some number of swap operations such that:

Each container contains only fruits of the same type.
No two fruits of the same type are located in different containers.

Function Description

organizingContainers has the following parameter(s):

int containter[n][m]: a two dimensional array of integers that represent the number of balls of each color in each container

Constraints:
$1 \leq q \leq 10$
$1 \leq n \leq 100$
$0 \leq containers[i][j] \leq 10^9$

Input Format:
The first line contains an integer 'q', the number of queries.

Each of the next 'q' sets of lines is as follows:

The first line contains an integer 'n', the number of containers (rows) and ball types (columns).
Each of the next 'n' lines contains 'n' space-separated integers describing row containers[i].

Output Format:

For each query, print Possible on a new line if David can satisfy the conditions above for the given matrix. Otherwise, print Impossible.

```
#include <stdio.h>

#include <stdlib.h>

void insertionSort(long int *p,long int n);

void asd();

int main(){

   asd();

 return 0;

}

void asd()


{

 int q;

 scanf("%d",&q);

 while(q--){

 int n,i,j;

 scanf("%d",&n);

 int M[n][n];

 long int *r,*c,*arr;

 arr=(long int *)malloc(n*n*sizeof(long int));

 *arr=n;

 r=(long int *)malloc(n*sizeof(long int)); c=(long int *)malloc(n*sizeof(long int));

 for(i=0;i<n;i++){

 for(j=0;j<n;j++){

 scanf("%d",&M[i][j]);

 r[i]+=M[i][j];

 c[j]+=M[i][j];

 }

 }

 int count=0;
```

```
for(i=0;i<n;i++){
for(j=0;j<n;j++){
if(r[i]==c[j])
{
count++;
break;
}
}
}
if(count==n)
printf("Possible\n");
else
printf("Impossible\n");
}
}
```

APPU needed a laptop, so he went to a neighbouring garage sale.

At a sale, there were n laptops.

The cost of a laptop with index i is ai rupees.

Some laptops have a negative price, meaning their owners are willing to pay APPU if he buys their broken laptop. APPU is free to purchase whichever laptop he desires.

Despite his strength, he can only handle m Laptops at a time, and he has no desire to return to the auction.

Please, help APPU find out the maximum sum of money that he can earn.

Constraints:

$1 \leq T \leq 10$

$1 \leq n,m \leq 100$

$-1000 \leq ai \leq 1000$

Input Format:

First line of the input contains T denoting the number of test cases.Each test case has 2 lines :

first line has two spaced integers n m.

second line has n integers [a0...ai...an-1].

Output Format:

The maximum sum of money that LALU can earn, given that he can carry at most m Laptops.

```
#include<iostream>
using namespace std;
int MEGA_SALE(int [],int ,int ) ;
void bubble_sort(int [],int ) ;
int minof(int ,int ) ;
int main()
 {
 int t,arr[100],no,i,k ;
 cin>>t ;
 while(t--)
 {
    cin>>no ;
    cin>>k ;
    for(i=0;i<no;i++)
       cin>>arr[i] ;

    no=MEGA_SALE(arr,no,k) ;
    cout<<abs(no)<<endl ;
 }
 return 0;
}
int MEGA_SALE(int arr[],int no,int k)
{
   int i ;
   bubble_sort(arr,no) ;
```

```
   int sum=0 ;
   for(i=0;i<k;i++)
      sum=minof(sum,sum+arr[i]) ;

   return sum ;
}
void bubble_sort(int arr[],int no)
{
   int i,j,temp ;
   for(i=0;i<no-1;i++)
   {
      for(j=0;j<no-i-1;j++)
      {
         if(arr[j]>arr[j+1])
         {
            temp=arr[j] ;
            arr[j]=arr[j+1] ;
            arr[j+1]=temp ;
         }
      }
   }
}
int minof(int a,int b)
{
   return a>b?b:a ;
}
```

Sajid is an third year student in a reputed institution.

Although he scored well in many subjects, he did not an expert in computer programming languages.

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

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

He collected previous year's questions. one of the repeated questions is to sort the given set of numbers using Selection Sort

The first line of the input contains the number of elements N, the second line of the input contains the numbers $\mathbf{A_i}$ to be sorted.

In the output print the the final sorted array in the given format.

Can you help him ?

Constraints

$1 <= \mathbf{N} <= 10^5$
$1 <= \mathbf{A_i} <= 10^9$

Input:
The first line of the input contains the number of elements

the second line of the input contains the numbers to be sorted.

Output:
print the the final sorted array in the given format.

```
#include <bits/stdc++.h>

using namespace std;

void dothis(){

   printf("void selectionSort(int arr[],int n)void swap(int *xp,int *yp)void printArray(int arr[],int size)");

}

int main()

{

   int n;

   cin>>n;

   vector<int>v(n) ;

   for (int i = 0; i < n; i++) {

      cin>>v[i];

   }
```

```
   sort(v.begin(),v.end());
   for (int i = 0; i < n; i++) {
      cout<<v[i]<<' ';
   }
   cout<<endl;


         return 0;
}
```

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:
$1 \leq T \leq 100$.
$1 \leq B \leq 10^5$.
$1 \leq Ai \leq 1000$, for all i.
$1 \leq N \leq 10^5$.

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 <bits/stdc++.h>
using namespace std;
int main()
{
   int t;cin>>t;
   while(t--){
      int n,tot,now=0;cin>>n>>tot;
      std::vector<int>v(n);
      for (int i = 0; i < n; i++) {
         cin>>v[i];
      }
      sort(v.begin(),v.end());
      for (int i = 0; i < n; i++) {
         now+=v[i];
         if(now>tot){
            cout<<i<<endl;
            break;
         }
      }
   }
          return 0;
          printf("void heapsort(int x[],int n)void makeheap(int x[],int n)heapsort(a,n);
makeheap(a,n);");
}
```

Sathya is a IT expert who training youngsters struggling in coding to make them better.

Sathya usually gives interesting problems to the youngsters to make them love the coding.

One such day Sathya provided the youngsters to solve that the given an array of integers and the numbers k1 and k2, get the sum of the two numbers.

Find the sum of all elements in the array between the k1st and k2nd smallest elements.

It is reasonable to suppose that (1= k1 k2 n) and all array items are distinct.

Constraints:

1<= T <= 100

1<= k1< K2 <= N <=50

Input Format:

The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains an integer N, denoting the length of the array.

Next line contains N space separated integers of the array.

Third line contains two space separated integers denoting k1'th and k2'th smallest elements.

Output Format:

For each test case in a new line output the sum of all the elements between k1'th and k2'th smallest elements.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   int t,i;cin>>t;
   while(t>0){
      int n,k1,k2,ans=0;
      cin>>n;
      int arr[n];
      for(i=0;i<n;i++) {
         cin>>arr[i];
      }
      cin>>k1>>k2;
      sort(arr,arr+n);
      for (int i = k1; i < k2-1; i++) {
```

```
        ans+=arr[i];
    }
    cout<<ans<<endl;
    t--;
  }
        return 0;
        printf("for(int i=0;i<n-1;i++)");
}
```

Selvan studies, engineering as per his father's wishes, while Aaron, whose family is poor, studies engineering to improve his family's financial situation. sumanth, however, studies engineering of his simple passion for developing data structure applications.

sumanth is participating in a hackathon for data structure application development.

sumanth task is to use Insertion Sort to sort the supplied set of numbers.

As a result, The input provides the number of components on the first line and the numbers to be sorted on the second line. Print the array's state at the third iteration and the final sorted array in the supplied format in the output.

Judge will determine whether the outcome is correct or not.

Can you help him ?

Constraints

1 <= N <= 10^5

1 <= Ai <= 10^9

Input Format:

The first line of the input contains the number of elements

the second line of the input contains the numbers to be sorted.

Output Format:

Fist line indicates print the status of the array at the 3rd iteration

second line print the the final sorted array in the given format.

```
#include <stdio.h>


void printArray(int arr[],int n){
   int i;
   for ( i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");
}
void insertionSort(int arr[],int n){
   int i, key, j;
   for (i = 1; i < n; i++) {
      if(i==3){
         printArray(arr,n);
      }
      key = arr[i];
      j = i - 1;
      while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}
int main()
{
   int n,i;
   scanf("%d",&n);
   int arr[n];
   for (i = 0; i < n; i++) {
      scanf("%d",&arr[i]);
```

```
    }
    insertionSort(arr, n);
    printArray(arr,n);
        return 0;
}
```

Simon work with Greek squares and matrix traces.

The trace of a square matrix is the sum of the values on the main diagonal (which runs from the upper left to the lower right).

An B-by-B square matrix is a Greek square if each cell contains one of B different values, and no value is repeated within a row or a column. In this problem, we will deal only with "beautiful Greek squares" in which the B values are the integers between 1 and B.

Given a matrix that contains only integers between 1 and B, we want to compute its trace and check whether it is a beautiful Greek square. To give some additional information, instead of simply telling us whether the matrix is a beautiful Greek square or not, show the number of rows and the number of columns that contain repeated values.

Constraints:
$1 \leq T \leq 100$.
$2 \leq B \leq 100$.
$1 \leq A_{i,j} \leq B$, for all i, j.

Input Format:
The first line of the input gives the number of test cases, T. T test cases follow. Each starts with a line containing a single integer B: the size of the matrix to explore. Then, B lines follow. The i-th of these lines contains B integers Ai,1, Ai,2 ..., Ai,B. Ai,j is the integer in the i-th row and j-th column of the matrix.

Output Format:
Print the output in a single lines contains the number of rows and the number of columns that contain repeated values.

In Sample Case #1, the input is a natural Latin square, which means no row or column has repeated elements. All four values in the main diagonal are 1, and so the trace (their sum) is 4.

In Sample Case #2, all rows and columns have repeated elements. Notice that each row or column with repeated elements is counted only once regardless of the number of elements that are repeated or how often they are repeated within the row or column. In addition, notice that some integers in the range 1 through **N** may be absent from the input.

In Sample Case #3, the leftmost and rightmost columns have repeated elements.

```
#include <bits/stdc++.h>
using namespace std;
#define lli long long int
void ss()
{
return;
cout<<" for(i=0;i<n;i++) void solve() int g[105][105];";
}
int main()
{
int t;
cin >> t;
for (int test_case = 1; test_case <= t; test_case++) {
int n;
cin >> n;
int** arr = (int**)(malloc(sizeof(int*) * n));
lli sum = 0;
int row = 0;
for (int i = 0; i < n; i++) {
arr[i] = (int*)(malloc(sizeof(int) * n));
map<int, int> mymap;
bool flag = false;
for (int j = 0; j < n; j++) {
cin >> arr[i][j];
if (i == j)
sum += arr[i][j];
if (flag == false && mymap.find(arr[i][j]) != mymap.end()) {
row++;
flag = true;
}
if (flag == false)
```

```
mymap[arr[i][j]]++;
}
}
int col = 0;
for (int i = 0; i < n; i++) {
map<int, int> mymap;
bool flag = false;
for (int j = 0; j < n; j++) {
if (flag == false && mymap.find(arr[j][i]) != mymap.end()) {
col++;
flag = true;
break;
}
if (flag == false)
mymap[arr[j][i]]++;
}
}
cout<< sum << " " << row << " " << col << endl;
}
return 0;
}
```

saravanan with his friends going to the theatre for a movie.

The seating arrangement is triangular in size.

Theatre staffs insisted the audience to sit in odd row if the seat number is odd and in even row if the seat number is even.

But the instruction is very confusing for saravanan and his friends.

So help them with the seating layout so that they can sit in correct seats.

Constraints:

$4 \leq N \leq 20$

Input Format:
Only line of input has single integer value representing the number of rows in the theatre.

Output Format:

Print the layout based on the number of rows specified in input.

Refer sample testcases for format specification.

```
#include <stdio.h>

int main()

{

   int i,j,N,k;

   scanf("%d", &N);

   for(i=1; i<=N; i++) {

      if(i%2==1) {

         k=-1;

      }

      else {

         k=0;

      }

      for(j=1; j<=i; j++) {

         printf("%d ",k+=2);

      }

      printf("\n");

   }

   return 0;

}
```

For some reason, your school's football team has chosen to spell out the numbers on their jerseys instead of using the usual digits. Being great fans, you're going to be ready to cheer for your favorite players by bringing letter cards so you can spell out their number. Each fan has different favorites, so they each need to bring different sets of letters.

The English spellings for the numbers 0 to 12 are:
ZERO ONE TWO THREE FOUR FIVE SIX
SEVEN EIGHT NINE TEN ELEVEN TWELVE

Input Format:

Read a set of integers from 0 to 12, separated by spaces, representing one fan's favorite players. The last integer will be 999, marking the end of the line.

Output Format:
Print the same numbers, then a period and a space. Then, in alphabetical order, print all the letters the fan needs to be able to spell any one of the jersey numbers provided

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

   int curr;

   multiset<char> mp;

   string names[] =
{"","ONE","TWO","THREE","FOUR","FIVE","SIX","SEVEN","EIGHT","NINE","TEN","ELEVEN","TWELVE"
};

   while(cin>>curr){

      if(curr==999){

         cout<<"0999"<<'.'<<' ';

         break;

      }

      cout<<curr<<' ';

      if(curr>12)continue;

      string now = names[curr];

      for(auto ch:now){

         mp.insert(ch);

      }

   }

   for (auto ch : mp) {

     cout<<ch<<' ';

   }


         return 0;

         printf("char nums[13][256]for(n=0;n<26;n++)");
```

}

There are K nuclear reactor chambers labelled from 0 to K-1. Particles are bombarded onto chamber 0. The particles keep collecting in the chamber 0.

However if at any time, there are more than N particles in a chamber, a reaction will cause 1 particle to move to the immediate next chamber(if current chamber is 0, then to chamber number 1), and all the particles in the current chamber will be be destroyed and same continues till no chamber has number of particles greater than N.

Given K,N and the total number of particles bombarded (A), find the final distribution of particles in the K chambers. Particles are bombarded one at a time. After one particle is bombarded, the set of reactions, as described, take place.

After all reactions are over, the next particle is bombarded. If a particle is going out from the last chamber, it has nowhere to go and is lost.

Constraints:

A will be between 0 and 1000000000 inclusive.

N will be between 0 and 100 inclusive.

K will be between 1 and 100 inclusive.

Input Format:

The input will consist of one line containing three numbers A,N and K separated by spaces.

All chambers start off with zero particles initially.

Output Format:

Consists of K numbers on one line followed by a newline.

The first number is the number of particles in chamber 0,

The second number is the number of particles in chamber 1 and so on.

```
#include <stdio.h>
int main()
{int a,n,k,i,b;
scanf("%d%d%d",&a,&n,&k);
for(i=0;i<k;i++){
   b=a%(n+1);
   printf(" %d",b);
```

```
   a=a/(n+1);
}
while(a>0){
   printf("a[100] a[r]");
}
   return 0;
}
```

Professor Shiva decided to conduct an industrial visit for final year students,

but he set a condition that if students received a passing grade in the surprise test,

they would be eligible to go on the industrial visit.

He asked the students to study a topic linked list for 10 minutes before deciding to conduct a surprise test.

Professor-mandated questions, such as the deletion of nodes with a certain data D, are now being asked.

For example

if the given Linked List is 5->10->15->10->25 and delete after 10 then the Linked List becomes 5->15->25.

Constraints

1< N < 100

1< D < 1000

Input Format
First line contains the number of datas- N.

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

Next line indicates the node data D that has to be deleted.

Output Format
Single line represents the linked list after required elements deleted.

```
#include <bits/stdc++.h>
```

```
using namespace std;

void ss()

{

return;

}

int main()

{

int n;

cin>>n;

int arr[n];

for (int i = 0; i < n; ++i)

{

cin>>arr[i];

}

int m;

cin>>m;

cout<<"Linked List:";

for(int p : arr)

{

if(p != m)

cout<<"->"<<p;

}

return 0;

cout<<"struct node node *next; void create() p2=p2->next; void del()";

}
```

A long time ago, there was a desolate village in India. The ancient buildings, streets, and businesses were deserted. The windows were open, and the stairwell was in disarray. You can be sure that it will be a fantastic area for mice to romp about in! People in the community have now chosen to provide high-quality education to young people in order to further the village's growth.

As a result, they established a programming language coaching centre. People from the coaching centre are presently performing a test. Create a programme for the GetNth() function, which accepts a linked list and an integer index and returns the data value contained

in the node at that index position.

Example

Input: 1->10->30->14, index = 2
Output: 30
The node at index 2 is 30

Constraints

1< N < 1000

1< X < 1000

1 < I < 1000

Input Format
First line contains the number of datas- N.
Second line contains N integers(the given linked list).
Third Line Index I

Output Format
First Line indicates the linked list

second line indicates the node at indexing position

```
#include <iostream>

using namespace std;

struct node

{

int data;

struct node *next;

}*head = NULL;

int i = 0,n;

int GetNth(struct node* head,int index)

{

while(n-i != index)

{

head = head -> next;
```

```
i++;
if(i == index) break;
}
return head -> data;
}
void display(node *u)
{
if(u == NULL) return;
display(u -> next);
cout<<"-->"<<u -> data;
}
int main()
{
int first;
cin>>n>>first;
node *t = new node;
t -> data = first;
t -> next = NULL;
head = t;
for(int i = 0; i< n-1 ; i++)
{
cin>>first;
node *u = new node;
u -> data = first;
u -> next = NULL;
t -> next = u;
t = u;
}
int index;
cin>>index;
node *hh = head;
```

```
cout<<"Linked list:";

display(head);

cout<<"\nNode at index="<<index<<":"<<GetNth(hh,index);

return 0;

}
```

Dr.Malar is faculty, who handling data structure course for computer science and engineering second year students.

one day this faculty was handling very interesting topic in data structure such that Linked List, she 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 making continuous disturbance by making unwanted noise.

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

the question was given to last bench students that is,

The new node is added at given position P of the given Linked List.

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

we add an item 30 at Position 3,

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 P and then insert the node.

Constraints

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 the position P where the node to be inserted. Fourth line contain the node X to be inserted.

Output Format

Single line represents the final linked list

```
 #include <bits/stdc++.h>

using namespace std;

struct node

{

int data;

struct node *next;

}*head = NULL;

int n;

int in_pos(int n)

{

int data1;

cin>>data1;

int i =1;

struct node *r = head;

while(i != n-1)

{

r = r-> next;

i++;

}
```

```
node *tt = new node;
tt -> data = data1;
tt -> next = r -> next;
r -> next = tt;
node *s = head;
cout<<"Linked List:";
while(s != NULL)
{
cout<<"->";
cout<<s-> data;
s = s-> next;
}
return data1;
}
void create()
{
int n;
cin>>n;
struct node *p = new node;
int __n;
cin>>__n;
p -> data = __n;
head = p;
int i;
for(i=0;i<n-1;i++)
```

```
{

int a;

cin>>a;

struct node *q = new node;

q -> data = a;

p -> next= q;

p = p->next;

}

p -> next = NULL;

}

int main()

{

create();

int r;

cin>>r;

int s = in_pos(r);

return 0;

cout<<s<<"for(i=0;i<n;i++)";

}
```

Kapildev works in the mobile phone marketing industry.

For example, if someone successfully answers this question, they will be given a mobile phone at a 50% discount.

One of the competition's requirements was to write a C programme that swapped nodes for two specified keys in a linked list with two keys.

By altering linkages, nodes should be switched.

When data consists of several fields, swapping data across nodes might be costly.

It is reasonable to presume that all keys in a linked list are unique.

example :

Given linked list : 10->15->12->13->20->14 and

swap keys X=12 and Y=20.

Linked list after swapping : 10->15->20->13->12->14
(if X or Y or Both are not present in Linked List, ABORT the Swapping)

Constraints

1< N < 1000

1< X < 1000

Input Format
First line contains the number of datas- N.
Second line contains N integers(the given linked list).
Third Line contains 2 key nodes(X and Y) to be Swapped.

Output Format
linked list before swapping keys
linked list after swapping keys

```
#include <iostream>

using namespace std;

struct node

{

int data;

struct node *next;

}*head = NULL;

void display(node *ss)

{

if(ss == NULL) return;
```

```
display(ss -> next);
cout<<"-->"<<ss -> data;
}
void swapNodes(struct node **head_ref,int x,int y)
{
if (x == y)
return;
node *prevX = NULL, *currX = *head_ref;
while (currX && currX->data != x) {
prevX = currX;
currX = currX->next;
}
node *prevY = NULL, *currY = *head_ref;
while (currY && currY->data != y) {
prevY = currY;
currY = currY->next;
}
if (currX == NULL || currY == NULL)
return;
if (prevX != NULL)
prevX->next = currY;
else
*head_ref = currY;
if (prevY != NULL)
prevY->next = currX;
```

```
else
*head_ref = currX;
node* temp = currY->next;
currY->next = currX->next;
currX->next = temp;
}
void create()
{
int n;cin>>n;
int rr;cin>>rr;
node *tt = new node;tt -> data = rr;
tt -> next = NULL;head = tt;
int i;
for(i=0;i<n-1;i++)
{
int a;
cin>>a;
node *q = new node;
q -> data = a;
q -> next = NULL;
tt -> next = q;
tt = q;
}
}
int main()
```

```
{create();

cout<<"before Swapping:";

display(head);

int x,y;

cin>>x>>y;

swapNodes(&head,x,y);

cout<<"\nafter Swapping:";

display(head);

return 0;

}
```

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:

$1 \leq N \leq 10^5$

$1 \leq K \leq N$

$1 \leq Ai \leq 10^9$

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 <bits/stdc++.h>
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;

}
```

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 <bits/stdc++.h>

#include<iostream>

#include<string.h>

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;

}
```

Arumugam is in the process of reorganising her library. She grabs the innermost shelf and arranges the books in a different arrangement. She shatters the shelf's walls. There will be no shelf barriers and simply books in the end. Make a printout of the book order.

Opening and closing walls of shelves are shown by '/' and '\' respectively whereas books are represented by lower case alphabets.

Constraints:

$2 \leq |S| \leq 10^3$

Input format

The first line contains string s displaying her library.

Output format

Print only one string displaying Arumugam library after rearrangement.

Note

The first character of the string is '/' and the last character of the string is '\' indicating outermost walls of the shelf.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

   string s,temp="";
```

```
    cin>>s;
    stack<string> stk;
    for (unsigned int i = 0; i < s.size(); i++) {
        if(s[i]==47||s[i]==92){
            if(!temp.empty()){
                stk.push(temp);
                temp.clear();
            }
        }
        else{
            temp.push_back(s[i]);
        }
    }
    while(!stk.empty()){
        cout<<stk.top();
        stk.pop();
    }


            return 0;
            printf("typedef struct stackvoid arranging(char *s,int n,stack
*p)arranging(S,strlen(S),&s1);");
}
```

Hassan enjoys jumping from one building to the next. However, he merely jumps to the next higher building and stops when there are none accessible. The amount of stamina necessary for a voyage is equal to the xor of all the heights Hassan leaps till he comes to a halt.

If heights are [1 2 4], and he starts from 1, goes to 2 stamina required is 1⊕2=3, then from 2 to 3. Stamina for the entire journey is 1⊕2⊕4=7. Find the maximum stamina required if can start his journey from any building.

Constraints

1≤ N ≤10^5

1 ≤ Hight ≤10^9

Input

First line: N, no of buildings.

Second line: N integers, defining heights of buildings.

Output

Single Integer is the maximum stamina required for any journey.

```
#include <stdio.h>
int main() {
 int i, j, arr[1000000], n, temp=0,st[1000000]= {0};
 scanf("%d",&n);
 for(i=0;i<n;i++){
 scanf("%d",&arr[i]);
 }
 st[n-1] = arr[n-1];
 temp = arr[n-1];
 for(i=n-2;i>=0;i--) {
 for(j=i+1;j<n;j++)
 if(arr[i]<arr[j]) {
 st[i]=arr[i]^st[j];
 break;
 }
 if(st[i] == 0)
```

```
st[i] = arr[i];

if(st[i] > temp)

temp = st[i];

}

printf("%d",temp);

return 0;

}
```

You are given an array A of Q integers and Q queries. In each query, you are given an integer i (1≤i≤N).

Your task is to find the minimum index greater than i (1≤i≤N) such that:

1. Sum of digits of Ai is greater than the sum of digits of Aj
2. Ai < Aj

If there is no answer, then print **-1**.

Constraints

1 < = N, Q ≤ 10^5

1 ≤ Ai ≤ 10^9

1 ≤ Qi ≤N

Input format

- The first line contains two numbers N and Q.
- The next line contains N numbers.
- Next Q lines contain Q queries.

Output format

Print the answer as described in the problem

```
#include<bits/stdc++.h>

using namespace std;

int main(){
```

```
int n,q;
cin>>n>>q;
int *a=new int [n];
for(int i=0;i<n;i++){
cin>>a[i];
}
int *arr=new int[n];
for(int i=0;i<n;i++){
int z=a[i];
int sum=0;
while(z>0){
sum+=(z%10);
z=z/10;
}
arr[i]=sum;
}
while(q--){
int Q;
cin>>Q;
int ans=-1;
for(int i=Q;i<n;i++){
if(a[i]>a[Q-1] && arr[i]<arr[Q-1]){
ans=i+1;
break;
}else{
```

```
continue;

}

}

cout<<ans<<' ';

}

return 0;

cout<<"if(arr[x]<arr[y]) if(arr2[x]>arr2[y]) ";

}
```

Umesh is an DS expert training youngsters struggling in DS to make them better.

Umesh usually gives interesting problems to the youngsters to make them love the DS.

One such day Umesh provided to the youngsters to solve the task such that, Reverse a Queue, Queue data structures work on the FIFO architecture so the element that has entered first in the list will go out from the list first.

Youngsters were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

Function Description

- INITIALIZE FRONT AND REAR = -1
- CALL ENQUEUE FUNCTION WHILE(I<GIVEN SIZE OF QUEUE)
- CALL REVERSE
- MAKE A FOR LOOP(I=FRONT,J=REAR;I<J;I++;J++)
- SWAP(FRONT,REAR)

Constraints:

0<size <100

0<data<1000

Input format:

First line indicates the size of the queue

Second line indicates the elements of the queue.

Output Format:

first line indicates the enqueue of each elements

last line indicates the reversed queue elements.

```
#include <stdio.h>
#define SIZE 100
void enqueue(int,int);
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;
 // 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]);

 }

}
```

You own a club on eerie planet. The day on this planet comprises of H hours.

You appointed C crew members to handle the huge crowd that you get, being the best club on the planet.

Each member of the crew has a fixed number of duty hours to work.

There can be multiple or no crew members at work at any given hour of the day.
Being on a weird planet, the rules of this club cannot be normal.

Each member of the crew only allows people who are taller than him to enter the club when he is at work.
Given the schedule of work and the heights of the crew members, you have to answer Q queries.

Each query specifies the time of entry and height of a person who is visiting the club.

You have to answer if the person will be allowed to enter the club or not.

Constraints:
$1 \leq H \leq 109$
$1 \leq C \leq 105$
$1 \leq Q \leq 105$
$1 \leq Si \leq Ei \leq H$
$1 \leq ti \leq H$
$1 \leq hi \leq 107$

Input:
The first line of the input contains 3 integers, H,C,Q. Representing a number of hours in a day, a number of crew members, and a number of queries respectively.
Next, C lines follow, where each line contains 3 integers, hi,Si,Ei, representing the height of the crew member and the start and end hour of his/her work schedule. He/she works for hours [Si,Ei], both inclusive.
Next, Q lines follow, each containing 2 integers, hi,ti, representing height and time (in an hour) of the person trying to enter the club.

Output:
Q lines, each line containing "YES" or "NO", without the quotes, answering if the person will be allowed to enter the club or not.

```
#include<stdio.h>
int main()
{
 long long int i,j,t,H,C,height,Q,S[100000],E[100000],h[100000];
 long long int nc=0,val=0,flag=0,maximum_height=0;
 scanf("%lld%lld%lld",&H,&C,&Q);

 for(i=0;i<C;i++)
 {
 scanf("%lld%lld%lld",&h[i],&S[i],&E[i]);
 if(h[i]>maximum_height)
 maximum_height=h[i];
 }
 for(i=0;i<Q;i++)
 {
 scanf("%lld%lld",&height,&t);
 if(height>maximum_height)
 printf("YES\n");
 else{
 val=0;
 nc=0;
 flag=0;
 for(j=0;j<C;j++)
 {
 if(t>=S[j] && t<=E[j])
```

```
{
nc++;
if(height<=h[j])
{
printf("NO\n");
flag=1;
break;
}
else
val++;
}
}

if(nc==val)
printf("YES\n");
else
if(flag==0)
printf("NO\n");
}

}
return 0;
printf("void enqueue(long long h,long long start,long long end) while(c--)");
}
```

Joe root is a Placement trainer. he is working as CDC trainer in reputed institution that during training the youngsters are struggling in queue concept.

Joe root usually gives interesting problems to the students to make them love the DS.

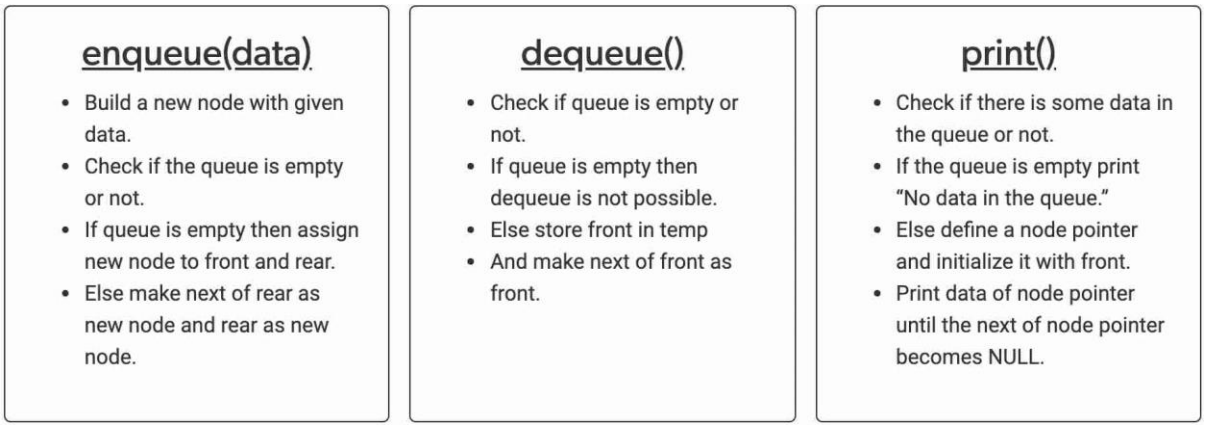
One such day Joe root provided to the final year students to solve the task such that, Queue implementation with arrays as using linked list for implementing queue and delete an element from the queue using linked list concept.

Queue data structures work on the FIFO architecture so the element that has entered first in the list will go out from the list first.

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

Being an exciting youngster can you solve it?

Function Description



Constraints:

0<size <100

0<data<1000

Input format:

First line indicates the size of the queue

Second line indicates the elements of the queue.

Output Format:

First line indicates the inserted elements in queue using linked list concept

second line indicates the one element dequeued from the queue using linked list concept.

#include <stdio.h>

#include <stdlib.h>

```
struct node *front = NULL;
struct node *rear = NULL;
struct node
{
 int data;
 struct node *next;
};
void linkedListTraversal(struct node *ptr)
{
 //printf("Printing the elements of this linked list\n");
 while (ptr != NULL)
 {
 printf("%d ", ptr->data);
 ptr = ptr->next;
 }
}
void enqueue(int d)
{
 struct node* new_n;
 new_n = (struct node*)malloc(sizeof(struct node));
 if(new_n==NULL){
 printf("Queue is Full");
 }
 else{
 new_n->data = d;
```

```
 new_n->next = NULL;
 if(front==NULL){
 front=rear=new_n;
 }
 else{
 rear->next = new_n;
 rear=new_n;
 }
 }
}
int dequeue()
{
 int val = -1;
 struct node *ptr = front;
 if(front==NULL){
 printf("Queue is Empty\n");
 }
 else{
 front = front->next;
 val = ptr->data;
 free(ptr);
 }
 return val;
}
int main()
```

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

Sathya is an DS expert training youngsters struggling in DS to make them better.

Sathya usually gives interesting problems to the youngsters to make them love the DS.

One such day Sathya provided to the youngsters to solve the task such that, insert an element in a Queue in FIFO order

Youngsters were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

Function Description

1. Define the maximum size of queue and initialize front and rear as -1.
2. In the main function we will initialize two variables that will store the data and the size of the queue.
3. Accept the data that we want to enter in a queue using a for loop.
4. After accepting the data use enqueue() function to insert the data in a queue.
5. In this function return queue id full , if the value of rear is equal to max-1.
6. Else increase the value of rear by 1.
7. After this insert the data that we have accepted earlier.
8. Now display the data of the queue using a disp() function.
9. In this function if the value of front is equal to -1 then it means list is empty.
10. Else initialize a for loop from front to rear and print the data that queue is holding.

Constraints:

0<size <100

0<data<1000

Input format:

First line indicates the size of the queue

Second line indicates the elements of the queue.

Output Format:

every lines indicates the enqueue of each elements

last line indicates the fine queued elements.

```
#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]);

 }

}
```

You are given an n×n 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<bits/stdc++.h>

using namespace std;

#define rep(i,a,b) for (int i=a; i<b; ++i)

int dp[1005][1005];
```

```
int main(){
    int n,m; cin>>n>>m;
    rep(i,1,n+1){
        rep(j,1,n+1){
            char x; cin>>x;
            dp[i][j] = (dp[i-1][j] - dp[i-1][j-1]) + dp[i][j-1] + (x=='*');
        }
    }
    while(m--){
        int y1 , x1, y2, x2; cin>>y1>>x1>>y2>>x2;
        cout<<dp[y2][x2]+ dp[y1-1][x1-1] - dp[y2][x1-1] - dp[y1-1][x2]<<endl;
    }
    return 0;
    cout<<"for(i=1;i<=n;i++)";
}
```

There are n children, and each of them independently gets a random integer number of candies between 1 and k.

What is the expected maximum number of candies a child gets?

Constraints

- $1 \le n \le 100$
- $1 \le k \le 100$

Input

The only input line contains two integers n and k.

Output

Print the expected number rounded to six decimal places.

```
#include <bits/stdc++.h>

using namespace std;

int N, K;

double ans, a, b;

int main(){

 scanf("%d %d", &N, &K);

 for(int i = 1; i <= K; i++){

 a = b = 1.0;

 for(int j = 1; j <= N; j++){

 a *= (double) i / K;

 b *= (double) (i-1) / K;

 }

 ans += (a-b) * i;

 }

 printf("%.6f\n", ans);

 return 0;

 cout<<"double power(double a,int k)";

}
```

A forest is an undirected graph without cycles (not necessarily connected).

Mohana and john are friends in kerala, both of them have a forest with nodes numbered from 1 to n, and they would like to add edges to their forests such that:

- After adding edges, both of their graphs are still forests.
- They add the same edges. That is, if an edge (u,v) is added to Mohana's forest, then an edge (u,v) is added to john's forest, and vice versa.

Mohana and johns want to know the maximum number of edges they can add, and which edges to add.

Constraints:

$1 \le n \le 105$,

$0 \le m1$

$m2 < n$

$1 \le u,v \le n,\ u \ne v$

Input

The first line contains three integers n, $m1$ and $m2$ — the number of nodes and the number of initial edges in Mohana's forest and john's forest.

Each of the next $m1$ lines contains two integers u and v — the edges in Mohana's forest.

Each of the next $m2$ lines contains two integers u and v ($1 \le u,v \le n,\ u \ne v$) — the edges in Johns's forest.

Output

The first line contains only one integer h, the maximum number of edges Mohana and John can add.

Each of the next h lines contains two integers u and v ($1 \le u,v \le n,\ u \ne v$) — the edge you add each time.

If there are multiple correct answers, you can print any one of them.

```
#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

int fa[1005],fa2[1005],n,m1,m2;

int gf(int x,int *f){

	return f[x]==x?x:f[x]=gf(f[x],f);

}

int main(){

	cin>>n>>m1>>m2;

	for(int i=1;i<=n;i++)fa[i]=fa2[i]=i;

	for(int i=1,x,y;i<=m1;i++)cin>>x>>y,fa[gf(x,fa)]=gf(y,fa);
```

```
for(int i=1,x,y;i<=m2;i++)cin>>x>>y,fa2[gf(x,fa2)]=gf(y,fa2);

cout<<n-max(m1,m2)-1<<'\n';

for(int i=1;i<=n;i++){

        for(int j=i+1;j<=n;j++){

                if(gf(i,fa)!=gf(j,fa)&&gf(i,fa2)!=gf(j,fa2)){

                        cout<<i<<' '<<j<<'\n';

                        fa[gf(i,fa)]=gf(j,fa);

                        fa2[gf(i,fa2)]=gf(j,fa2);

                }

        }

}

return 0;

cout<<"while(m1--)";
```

}Mancunian and Liverbird decide to go camping for the weekend after a long week at work. They came upon an unusual tree with N nodes while walking through a forest. From 1 to N, the vertices are numbered.

A colour is allocated to each node in the tree (out of C possible colors). They decide to work together (for a change) and put their reasoning abilities to the test because they are bored. At vertex 1, the tree is rooted. They aim to locate the nearest ancestor with the same hue for each node.

Constraints

- $1 \leq N \leq 100,000$
- $1 \leq C \leq 100,000$

Input format

The first line contains two integers *N* and *C* denoting the number of vertices in the tree and the number of possible colors.
The second line contains $N-1$ integers. The i th integer denotes the parent of the i+1 th vertex.
The third line contains *N* integers, denoting the colors of the vertices. Each color lies between *1* and *C* inclusive.

Output format

Print *N* space-separated integers. The ith integer is the vertex number of lowest ancestor of the ith node which has the same color. If there is no such ancestor, print *1* for that node.

```
#include<bits/stdc++.h>

using namespace std;

int main() {

int n,i,c;

scanf("%d %d", &n, &c);

int tree[n+1][2];

tree[1][0] = -1;

for(i=2;i<=n;i++) {

scanf("%d", &tree[i][0]);

}

for(i = 1; i <= n; i++) {

scanf("%d", &tree[i][1]);

}

int parent;

for(i = 1; i<= n; i++) {

parent = tree[i][0];

while(parent != -1 && tree[parent][1] != tree[i][1]) {

parent = tree[parent][0];

}

printf("%d ", parent);

}
```

return 0;

}

A beautiful *code* of a tree of n nodes is a sequence of n−2 integers that uniquely specifies the structure of the tree.

The code is constructed as follows: As long as there are at least three nodes left, find a leaf with the smallest label, add the label of its only neighbour to the code, and remove the leaf from the tree.

Given a beautiful code of a tree, your task is to construct the original tree.

Constraints

- $3 \le n \le 2 \cdot 10^5$
- $1 \le a,b \le n$

Input

The first input line contains an integer n: the number of nodes. The nodes are numbered 1,2,…,n.

The second line contains n−2 integers: the beautiful code.

Output

Print n−1 lines describing the edges of the tree. Each line has to contain two integers a and b: there is an edge between nodes a and b. You can print the edges in any order.

```
#include <bits/stdc++.h>

using namespace std;

#define f(i,a,n) for(int i=a;i<n;i++)

#define X(a,b) if(a==b)

vector< int > vi;

const int maxN = 2e5+5;

int N, a[maxN], deg[maxN];

priority_queue<int, vector<int>, greater<int>> q;
```

```
int main(){
 scanf("%d", &N);
 fill(deg+1, deg+N+1, 1);
 //for(int i = 0; i < N-2; i++)
 f(i,0,N-2){
 scanf("%d", &a[i]);
 deg[a[i]]++;
 }
 //for(int i = 1; i <= N; i++)
 f(i,1,N+1)
 //if(deg[i] == 1)
 X(deg[i],1)
 q.push(i);
 f(i,0,N-2){
 int u = a[i];
 int v = q.top(); q.pop();
 deg[u]--; deg[v]--;
 //if(deg[u] == 1)
 X(deg[u],1)
 q.push(u);
 printf("%d %d\n", v, u);
 }
 //for(int i = 1; i <= N; i++)
 f(i,1,N+1)
 if(deg[i])
```

```
 printf("%d ", i);

}
```

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;
```

}

Byteland has n cities and m roads between them. The goal is to construct new roads so that there is a route between any two cities.
Your task is to find out the minimum number of roads required, and also determine which roads should be built.

Constraints

$1 \le n \le 10^5$
$1 \le m \le 2 \cdot 10^5$
$1 \le a,b \le n$

Input

The first input line has two integers n and m: the number of cities and roads. The cities are numbered 1,2,…,n.

After that, there are m lines describing the roads. Each line has two integers a and b: there is a road between those cities.

A road always connects two different cities, and there is at most one road between any two cities.

Output

First print an integer k: the number of required roads.

Then, print k lines that describe the new roads. You can print any valid solution.

```
#include <bits/stdc++.h>

using namespace std;

#define rep(i, a, b) for(int i = a; i < (b); ++i)

#define trav(a, x) for(auto& a : x)

#define all(x) begin(x), end(x)

#define sz(x) (int)(x).size()

typedef long long ll;

typedef pair<int, int> pii;

typedef vector<int> vi;

vi val, comp, z, cont;
```

```
int Time, ncomps;
template<class G, class F> int dfs(int j, G& g, F& f) {
 int low = val[j] = ++Time, x; z.push_back(j);
 trav(e,g[j]) if (comp[e] < 0)
 low = min(low, val[e] ?: dfs(e,g,f));

 if (low == val[j]) {
 do {
 x = z.back(); z.pop_back();
 comp[x] = ncomps;
 cont.push_back(x);
 } while (x != j);
 f(cont); cont.clear();
 ncomps++;
 }
 return val[j] = low;
}
template<class G, class F> void scc(G& g, F f) {
 int n = sz(g);
 val.assign(n, 0); comp.assign(n, -1);
 Time = ncomps = 0;
 rep(i,0,n) if (comp[i] < 0) dfs(i, g, f);
}
int main() {
 cin.sync_with_stdio(0); cin.tie(0);
```

```
cin.exceptions(cin.failbit);


int n, m;
cin >> n >> m;
vector<vi> g(n);
while(m--) {
int a, b;
cin >> a >> b;
a--, b--;
g[a].push_back(b);
g[b].push_back(a);
}
vi r;
scc(g, [&](vi &c) { r.push_back(c[0]); });
cout << sz(r)-1 << '\n';
rep(i, 1, sz(r))
cout << r[0]+1 << " " << r[i]+1 << '\n';
return 0;
}
```

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<bits/stdc++.h>

using namespace std;

void solve(){}

int32_t 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;
}
```

Canthi and Sami are having a game! The game is extremely similar to chess, but there is only one piece on the board, which is the Queen. In addition, Queen may only go to the top left corner.

For clarification, If Queen is placed at i,j then in a turn queen can move:
1) Any number of cells leftwards.
2) Any number of cells upwards.
3) Any number of cells Diagonally(only N-W direction).

Please note that board is quarter infinite i.e there is a top left corner but it extends to infinity in south and east direction..

Functional Description

1) Canthi will always play the first turn.
2) They both will get alternative turns.
3) They must move the queen in their turn (No skip option) and that too according to the rule specified above.
4) Whosoever is unable to make a move loses.

Given The position of queen on the board(0 index based). Print who will win the game.

Constraints:
1<=t<=10000
0<=a,b<=1000000

Input:
First line of input contains an integer t - no of test cases.
Each test case is described as a line containing two integers a and b. which is the position of queen on the board.

Output:
print the name of person who will win the game.

```
#include <stdio.h>

#include<math.h>

int v[2000000],i,t;

double fi;

int main()

{

   fi=((double)((1+sqrt(5))/2.0));

   for(i=1;i<=1000000;i++)

   v[i]=-1;

   for(i=1;i<=1000000;i++)

   v[(int)(fi*(double)i)] = (int)(fi*fi*i);

   scanf("%d",&t);

   while(t--){

      int a,b;

      scanf("%d %d",&a,&b);

      if(v[a]==b)

      printf("sami\n");

      else

      printf("canthi\n");
```

```
    }

    return 0;

}
```