Question 1

Problem Description:

Archana wants to decorate his house by balloons. She plans to buy exactly M ones. She can only buy them from Grace Super Market. There are only two kind of balloons available in that shop. The shop is very different. If you buy 'X' balloons of kind 1 then you must pay C x X^2 and D x Y^2 if you buy balloons of kind 2.

Please help Archana buys exactly M balloons that minimizes amount she pays.

Constraints:

1 <= T <=10^5

1 <= M, C, D <= 10^5

Input Format:

The first line contains T, denoting the number of test cases.

Each of test case is described in a single line containing three space-separated integers M, C, D.

Output Format:

Please help Archana buys exactly M balloons that minimizes amount she pays.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
string z = "while(M>0)";
int cost(int x, int y, int c, int d)
{
    return c * x * x + d * y * y;
}
int main()
{     int t,m,c,d;     cin>>t;     while(t--){         cin>>m>>c>>d;         i
            min_ = min(cost(oth, m-oth, c, d), min_);
        }
        cout << min_ << "\n";
    }
}
```

Question 2

Problem Description:

The Allies are trying to get a message 25 meters straight up a cliff to a waiting team in the cyberpunk virtual wars of 7702 (spoiler alert, it isn't going well). They are trying to build a contraption which will get a messenger up the cliff to deliver the message in person (mail is expensive in the year 7702, so . . . contraptions!)

Input Format:

You will receive, on a single line separated by spaces, the name of the contraption the Allies want to build, the speed to which it can launch something into flight and the distance per time unit that the contraption can launch something. You may receive the following units: MILES, KILOMETERS, YARDS, FEET, METERS, INCHES, CENTIMETERS, HOUR, MINUTE, SECOND.

Output Format:

Calculate (using the formula given below) the height to which the contraption can launch the object (ignoring the weight of the messenger ... being virtual, their weight is measured in photons), in meters rounded to the nearest one hundredth. Print the name of the contraption and how high it will launch our messenger using the format given below. If it will reach at least 25.00 meters, print afterwards: SUCCESS, else print: SPLAT. However there is a roof over the tunnel in the cliff the Allies are aiming for starting 50 meters up. If the messenger will be launched higher than 50 meters, print OUCH (because they will hit the roof).

Explanation:

$$\text{height} = \frac{speed^2}{2(gravity)} = \frac{speed^2}{2(\frac{9.805m}{s^2})}$$

- 1 meter is equal to 3.28 feet for the purposes of this problem

Because the units for the acceleration of gravity are given in meters/second, and because your output needs to be in meters, your first step needs to be converting your input rate of speed to meters/second (if it isn't already in meters/second). After that simply square the speed and then divide by 2 times the acceleration of gravity (9.805m/s^2) to get the maximum vertical distance the contraption will launch our brave messenger.

In this Test case 1, 1980.00 feet / minute converts to 33 feet / second. Converting feet to meters (using the conversion of 3.28 m/f given above) then gives us 10.060975609756097560975609756098 meters / second (don't round anything until printing your final answer.)

Squaring that will give us 101.22323022010707911957168352171 (m/s)^2. We then divide that by 2* (9.805m/s^2).

That gives us a distance of 5.161816941361911224863420883310 2 meters traveled, which we will round to the nearest hundredth giving us 5.16, which is TOO SHORT, so our messenger will hit the cliff wall, so we print SPLAT! after our output.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
void solve(){    cout<<"break;";  }
int main(){
    string s1,s2,s3,s4;
    double r;
    double h;
cin>>s1>>r>>s2>>s3>>s4;
    if(s2=="FEET")
    r=r/3.28;
    //cout<<r<<endl;
```

```
    if(s2=="KILOMETERS") r=r*1000;
    if(s2=="YARDS") r=r*0.9144;
    if(s2=="INCHES") r=r*0.0254;
    if(s2=="MILES") r=r*1609.34;
    if(s4=="HOUR") r=r/3600;
    if(s4=="MINUTE") r=r/60;
    if(s2=="CENTIMETERS") r=r/100;
    h=r*r/(2*9.805);
cout<<s1<<" will launch the message "<<fixed<<setprecision(2)<<h<<" meters hig
    if(h>50) cout<<"OUCH!";
    else if(h<25) cout<<"SPLAT!";
    else cout<<"SUCCESS!";
        return 0; }
```

Question 3

Problem Description:

Mr Somu has another problem for Agi today. He has given him three positive integers B, N and R and wants him to calculate the remainder when B^N! is divided by R. As usual, N! denotes the product of the first N positive integers.

Constraints:

$1 \leq T \leq 100$

$1 \leq B \leq 10$

$1 \leq N \leq 10$

$1 \leq R \leq 10$

Input Format:

The first line of the input gives the number of test cases, T. T lines follow. Each line contains three integers B, N and R, as described above.

Output Format:

Print the output in a separate lines.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int t;
    cin>>t;
    while(t--){
        int b,n,r;
        cin>>b>>n>>r;
        int z=1;
        for(int i=1;i<=n;i++){
            z=z*i;
```

```
        }
        int res;
        res=pow(b,z);
        cout<<res%r<<endl;
    }
    return 0;
    cout<<"if(n%2==1)";
}
```

Question 4

Problem Description:

Given two integers: 'b' and 'a' and 'b' is divisible by 2a, you have to first write down the first 'b' natural numbers in the following form:

1. At first take first 'a' integers and make their sign negative

2. Then take next 'a' integers and make their sign positive

3. The next 'a' integers should have negative signs and continue this procedure until all the 'b' integers have been assigned a sign.

For example, let 'b' be 12 and 'a' be 3. Then we have -1 - 2 - 3 + 4 + 5 + 6 - 7 - 8 - 9 + 10 + 11 + 12. If b = 4 and a = 1, then we have -1 +2 -3 + 4.

Now your task is to find the summation of the numbers considering their signs.

Constraints:

$1 \leq T \leq 200$

$2 \leq b \leq 10^9, 1 \leq a$

And you can assume that b is divisible by 2*a.

Input Format:

The first line contains T, denoting the number of test cases.

Each case starts with a line containing two integers b and a.

Output Format:

Print the output in a separate lines contains to find the summation of the numbers considering their signs.

Program Code:

```
#include <iostream>
using namespace std;
int main()
{
    int t;
    long long m;
```

```
    long long n;
    long long ans;
    scanf("%d", &t);
    for (int cs = 1; cs <= t; cs++) {
        scanf("%lld %lld", &n, &m);
        ans = (n * m ) / 2;
        printf("%lld\n",ans);
    }

}
```

Question 6

Problem Description:

Trapped by a river and racing against time, our fearless heroes need to quickly cross it in order to stop mom from placing the wrong pizza order. (Funny story, turns out Greg was only joking about the anchovies).

Luckily, our heroes have found a ramp on their side of the river (what could go wrong?).

Input Format:

You will receive 4 lines of information:

Line 1: The name of the vehicle

Line 2: The length of the ramp (in meters, always a whole 32-bit integer)

Line 3: The acceleration rate of the vehicle (in meters/second squared, floating point decimal of max size 2147483647.0)

Line 4: The width of the river (in meters, floating point decimal of max size 2147483647.0)

Output Format:

Using that information, 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!

If the distance which can be covered is:

< the width of the river minus 5 meters, add SPLASH! to your output

>= river-width minus 5 meters AND <= river-width, add BARELY MADE IT!

> river-width, add LIKE A BOSS!

Explanation:

This will be a multi-equation calculation using the following data:

- ramp1 = Ramp length
- rate1 = Vehicle acceleration rate (Note, in this problem, no vehicle has a maximum speed, they can all accelerate indefinitely, because SCIENCE!)

You will first need to figure out at what time the vehicle will run out of ramp if accelerating at the constant rate given in the input. Solve for that time (in seconds) using:

- time1=sqrt(2.0*ramp1/rate1);

Once you have that time, plug that into the following formula to solve for the rate of speed the vehicle will be traveling at the moment it runs out of ramp:

- speed1=time1*rate1;

Now, with those 2 pieces of key information you can calculate the distance the vehicle will be be able to cover when launched diagonally to cross the river (under ideal conditions):

- distance=speed1*speed1/9.805;

So, in our test case 1, given a Motorcycle using a 100 meter long ramp, with a constant acceleration rate of 30m/s, we will get: * time1 = 2.58s until it reaches the end of the ramp * speed1 = 77.46m/s rate of speed when it leaves the ramp * 257.8 meters distance traveled

Disclaimer We are aware that this isn't fully accurate according to physics -- in the universe of this problem, the equations given are correct, and all that you need.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    string str;
    int ramp1;
    double rate1,wr;
    getline(cin,str);
    cin>>ramp1>>rate1>>wr;
    double time1,speed1,dist1;
    time1=sqrt(2.0*ramp1/rate1);
    speed1=time1*rate1;
    dist1=speed1*speed1/9.805;
    cout<<str<<" will reach a speed of "<<std::fixed<<std::setprecision(2)<<sp
    if(dist1<(wr-5))
    cout<<"SPLASH!";
    else if(dist1>wr)
    cout<<"LIKE A BOSS!";
    else
    cout<<"BARELY MADE IT!";
    return 0;
}
```

Question 7

Problem Description:

There is a Gangaroo initially placed at the origin of the coordinate plane. In exactly 1 second, the gangaroo can either move up 1 unit, move right 1 unit, or stay still. In other words, from position (a, b), the gangaroo can spend 1 second to move to:

- (a+1, b)
- (a, b+1)
- (a, b)

After T seconds, a farmer who sees the gangaroo reports that the gangaroo lies on or innner side a square of side-length 's' with coordinates (A, B), (A+s, B), (A, B+s), (A+s, B+s). Calculate how many points with integer coordinates on or innner side this square could be the gangaroo's position after exactly T seconds.

Constraints:

0 <= A, B <=200

1 <= s <= 200

0 <= T <= 500

Input Format:

The input line contains four space-separated integers: A, B, s, and T.

Output Format:

Print the output in a single line contain to find the number of points with integer coordinates that could be the gangaroo's position after T seconds.

Program Code:

```
#include <stdio.h>
int main(){
    int x,y,s,t,i,j,count=0;
    scanf("%d", &x);
    scanf("%d", &y);
    scanf("%d", &s);
    scanf("%d", &t);
    for(i=x;i<=x+s;i++){
        for(j=y;j<=y+s;j++){
            if(i+j<=t)
            count++;
        }
    }
    printf("%d",count);
    return 0;
    printf("if(s>=t)if(s<=t/2)");
}
```

Question 8

Problem Description:

Maari wish to buy watches from the famous online flipkart.

Usually, all watches are sold at the same price, 'p' rupees. However, they are planning to have the seasonal big billion days 2022 sale next month in which he can buy watches at a cheaper price.

Specifically, the first watch will amount 'p' rupees, and every subsequent watch will amount 'd' rupees less than the previous one. This continues until the amount becomes less than or equal to 'm' rupees, after which every watch will amount 'm' rupees. How many watches he can buy during the big billion days 2022 sale?

Constraints:

1 <= m <= p <= 100

1 <= d <= 100

1 <= s <= 10^4

Input Format:

The input line contains four space-separated integers p, d, m and s.

Output Format:

Print the output in single line contains to find the how many watches he can buy during the big billion days 2022 sale.

Program Code:

```
#include <iostream>
using namespace std;
int main()
{
    int p,d,m,s;
    cin>>p>>d>>m>>s;
    int sum=0,count=0;
    while(sum<=s)
    {         if(p<m)
               sum+=m;
               else
               sum+=p;
               p-=d;
               count++;
    }
    cout<<count-1;
    return 0;
    cout<<"while(p<=s)"; }
```

Question 9

Problem Description:

The Mask ate a block of dynamite to save Tina during Dorian's rampage -- now he has a taste for it.

Help the Mask figure out how much dynamite he can eat without being destroyed.

Input Format:

You will receive on a line 3 integers (or fractions) separated by spaces.

The first number will be the number of sticks of dynamite.

The second will be the size of the stick (1/4, 1/3, 1/2, 1, 2 or 3).

The third number will be the tensile limit (in Megajoules (MJ)) the Mask can tolerate at that moment, due to being tired, having suffered other damage, being freshly out of bed, etc.

Output Format:

Determine if the Mask can survive the energy output from the blast if he eats the dynamite. See explanation section for formula. Print the amount of explosive force the dynamite will produce (rounded to 2 decimal places), a space then if the force is <= the tensile limit the Mask can tolerate, print "the Mask can eat it!", else print "the Mask should not eat it!"

Explanation:

To determine the Megajoules (MJ) of explosive force, you will need the following information:

- One (1) whole stick of dynamite weighs 0.45 kilograms (kg).
- 7.5MJ of explosive force are released when 1kg of dynamite explodes.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    float a,c,d;
    string b;
    cin>>a>>b>>c;
    float res;
    int z=b.size();
    if(z==1)
    d=b[0]-48;
    else
    d=(float)(b[0]-48)/(b[2]-48);
    res=a*d*0.45*7.5;
    if(res>c){
        cout<<res<<" the Mask should not eat it!";
    }
    else
    cout<<fixed<<setprecision(2)<<res<<" the Mask can eat it!";
    return 0;
    cout<<"for";
}
```

Question 10

Problem Description:

Given 'm' positive integers denoting an upgrading map where the width of every one bar is 1, find how much water it can hold after Rainfall.

Example 1:

Figure 1

Explanation:

In Figure 1 upgrading map (black) is denoted by array 0 1 0 2 1 0 1 3 2 1 2 1. In this case, 6 units of water (blue) are being holded.

Constraints:

m == height.length

1 <= m <= 2 * 10^4

0 <= height[i] <= 10^5

Input Format:

First line contains an integer m.

Second line contains 'm' space separated integers representing the elevation map.

Output Format:

Print the output in a single line contains to find how much water it can hold after rainfall.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
#define f(n) for(i=0;i<n;i++)
#define g(n) for(i = 1; i < n; i++)
#define k(n) for(i=n-2;i>=0;i--)
int maxWater(int arr[], int n)
{
    int left[n],i;
    int right[n];
    int water = 0;
    left[0] = arr[0];
    g(n)
        left[i] = max(left[i - 1], arr[i]);
    right[n - 1] = arr[n - 1];
    k(n)
        right[i] = max(right[i + 1], arr[i]);
    for(i = 1; i < n-1; i++)
    {
      int var=min(left[i-1],right[i+1]);
      if(var > arr[i])
```

```
      {
        water += var - arr[i];
      }
    }
    return water;
}
int main()
{
    int n,i;
    cin>>n;
    int arr[n];
    f(n){
        cin>>arr[i];
    }
    cout << maxWater(arr, n);
    return 0;
}
```

Question 11

Problem Description:

RSA is a public key cryptosystem. Most important part of RSA is to choose two primes 'p' and 'q', and multiply them to construct an integer 'm'. You dont have to break RSA in this problem. Instead, given N, you have to find the number of integers less than equal to 'N' that are product of two primes. More formally, find |{p * q | p * q <= N, p, q are primes}|.

Constraints:

1 <= T <= 20

1 <= N <= 2*10^9

Input Format:

The first line contains T, the number of test cases. T test cases follow. Each test case consists of a single line containing integer N.

Output Format:

Print the number of integers below 'N' that are product of two primes.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
void solve(){
    cout<<"break;";
}
bool isProduct(int num)
{
    int cnt = 0;
    for (int i = 2; cnt < 2 && i * i <= num; ++i) {
        while (num % i == 0) {
            num /= i;
```

```
            ++cnt;
        }
    }
    if (num > 1)
        ++cnt;
    return cnt == 2;
}
void findNumbers(int N)
{
    vector<int> vec;
    for (int i = 1; i <= N; i++) {
        if (isProduct(i) ) {
            vec.push_back(i);
        }
    }
    cout<<vec.size()<<endl;
}
int main()
{
    int t,N;
    cin>>t;
    while(t--){
        cin>>N;
        findNumbers(N);
    }
    return 0;
}
```

Question 12

Problem Description:

Having conquered the world of extreme underwater basket weaving, you are now moving on to Xtreme Surface Skiing! As the most extremely experienced X-games competitor in your new league, they are seeking your expertise to decide what materials to use for the first round of competitions (before the shark jumping round).

Input Format:

You will receive on a single line a SKI MATERIAL ("X" coordinate in table below, A.K.A. the top row), followed by a skiing SURFACE ("Y" coordinate in table below, A.K.A. the first column). Use the material-to-surface lookup table in the discussion section to determine the coefficient of friction for the given ski material on the given surface. Use that coefficient of friction to find the minimum slope angle (using the inverse tangent (known as arctan)) the league would need to use for the competition.

Coefficients of Friction Lookup Table

There are 5 skiing surfaces and 3 ski materials.

SURFACE	SKI MATERIAL		
	RUBBER	WOOD	STEEL
CONCRETE	0.90	0.62	0.57
WOOD	0.80	0.42	0.30
STEEL	0.70	0.30	0.74
RUBBER	1.15	0.80	0.70
ICE	0.15	0.05	0.03

Output Format:

Output the coefficient of friction you found in the lookup table followed by a space and minimum slope the league would need to use for Xtreme skiing on those materials, rounded to the nearest whole tenth. You are guaranteed that all materials given as inputs will have corresponding outputs that are real, non-negative numbers.

Explanation:

In the Test Case 1, the SKI MATERIAL is WOOD, and the SKIING SURFACE is RUBBER. Looking up a SKI MATERIAL of WOOD and a SURFACE of RUBBER in our table, we find the coefficient of friction to be 0.80. Arctan(0.80) is 38.65980825 degrees. Rounding that up to the nearest tenth we get: 38.7.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
string z = "break; if";
int main(){
    map<string, int> surfaces {{"CONCRETE", 0}, {"WOOD", 1}, {"STEEL", 2}, {"R
    map<string, int> mats {{"RUBBER", 0}, {"WOOD", 1}, {"STEEL", 2}};
    float table[5][3] = {
        {0.9, 0.62, 0.57},
        {0.8, 0.42, 0.3},
        {0.7, 0.3, 0.74},
        {1.15, 0.8, 0.7},
        {0.15, 0.05, 0.03}
    };
    string a, b;
    cin>>a>>b;
    float z = table[surfaces[b]][mats[a]];
    float res = atan(z) * (180/3.14159);
```

```
    printf("%.2f %.1f", z, res);
}
```

Question 14

Problem Description:

Shankar is a volleyball trainer at government school in Madurai, he has been tasked with choosing a team of exactly P players to represent in that school. There are N players for his to choose from. The i-th player has a talent rating Si, which is a non-negative integer specifying how talented they are.

Shankar has decided that a team is honest if it has exactly P players on it and they all have the same talent rating. This is everyone plays in a single team. Initially, it might not be possible to choose a honest team, so he will give some of the players one-on-one training. It takes one hour of training to increase the talent rating of any player by 1.

The competition season is starting very soon (in fact, the first match has already started!), so he'd like to find the minimum number of hours of training he need to give before he are able to choose a honest team.

Constraints:

$1 \leq T \leq 150$.

$1 \leq Si \leq 1000$, for all i.

$2 \leq P \leq N$.

$2 \leq N \leq 10000$.

Input Format:

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

Next line containing the two integers N and P, the number of players and the number of players he need to choose, respectively. Then, another line follows containing N integers Si; the i-th of these is the talent of the i-th student.

Output Format:

Print the output in a separate lines contains to find the minimum number of hours of training he need to give before he are able to choose a honest team.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int N = (int) 1e6 + 6, mod = (int) 0;
int a[N];
long long sum[N];
int main() {
    int tc;
    cin >> tc;
    for (int tt = 1; tt <= tc; ++tt) {
```

```
        int n, p;
        cin >> n >> p;
        for (int j = 0; j < n; ++j)
            cin >> a[j];
        sort(a, a + n);
        int i;
        for(i=0;i<n;i++)
            sum[i + 1] = sum[i] + a[i];
        long long res = 1e18;
        for (int j = p - 1; j < n; ++j) {
            long long s = sum[j + 1] - sum[j - (p - 1)];
            long long cost = (LL) a[j] * p - s;
            res = min(res, cost);

        }
        cout << res << '\n';
    }
}
```

Question 16

Problem Description:

Ganesan has a string S consisting of lowercase English letters.

On this string, he will do the operation below just once.

- First, choose a non-negative integer K.
- Then, shift each character of S to the right by K (see below).

Here,

- a shifted to the right by 1 is b;
- b shifted to the right by 1 is c;
- c shifted to the right by 1 is d;
- …
- y shifted to the right by 1 is z;
- z shifted to the right by 1 is a.

For example, b shifted to the right by 4 is f, and y shifted to the right by 3 is b.

You are given a string T. Determine whether Ganesan can make S equal T by the operation above.

Constraints:

- Each of S and T is a string of length between 1 and 10^5 (inclusive) consisting of lowercase English letters.
- The lengths of S and T are equal.

Input Format:

S

T

Output Format:

If Ganesan can make S equal T, print Yes; if not, print No.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string s,s2;
    cin>>s>>s2;
    int z = s.length();
    int i;
    int a[z];
    for(i=0;i<(int)s.length();i++){
        a[i]=s[i+1]-s[i];
    }
    for(int i=0;i<z-2;i++){
        if(a[i]!=a[i+1]){
        cout<<"No";
        return 0;}
    }
    cout<<"Yes";
    return 0;
}
```

Question 17

Problem Description:

Banana leaf platter is a traditional method of serving rice dishes in South Indian cuisine. Due to the migration of South Indians, banana leaf rice can also be found in areas with significant ethnic South Indian diaspora such as Malaysia and Singapore.

Irfan is a banana leaf sales person.
he has N stacks of banana leafs.

Each stack contains K leafs.

Each leaf has a positive beauty value, describing how attractive it looks.

Irfan would like to take exactly P leafs to use for lunch today. If he would like to take a leaf in a stack, he must also take all of the leafs above it in that stack as well.

Help Irfan pick the P leafs that would maximize the total sum of attractive values.

Constraints:
$1 \leq T \leq 100$.
$1 \leq K \leq 30$.
$1 \leq P \leq N * K$.
$1 \leq N \leq 50$.

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 three integers N, K and P. Then, N lines follow. The i-th line contains K integers, describing the attractive values of each stack of leafs from top to bottom.

Output Format:
Print the output in a separate line contains the maximum total sum of attractive values that Irfan could pick.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ar array
void dummy(){}
int n, k, p, a[50][30];
int dp[51][1501];
void solve() {
cin >> n >> k >> p;
memset(dp, 0xc0, sizeof(dp));
dp[0][0]=0;
for(int i=0; i<n; ++i) {
memcpy(dp[i+1], dp[i], sizeof(dp[0]));
for(int j=0, s=0; j<k; ++j) {
cin >> a[i][j];
s+=a[i][j];
//use j+1 plates
for(int l=0; l+j+1<=p; ++l)
dp[i+1][l+j+1]=max(dp[i][l]+s, dp[i+1][l+j+1]);
}
}
cout << dp[n][p] << "\n";
}
int main() {
int n, i;
cin >> n;
for(i=0;i<n;i++) {
solve();
}
return 0;
cout<<"int max(int a,int b) for(int i = 0;i < n;i++) ";
}
```

Question 18

Problem Description:

Scrooge McDuck's vault is practically overflowing.

Normally, Scrooge wouldn't consider something like this a problem. But a recently passed city law in Duckburg mandates a minimum height for diving boards, and Scrooge's board is too close to the surface of his fortune.

The city will remove any boards that don't comply with the new rule. One of Scrooge's favorite activities is diving into his tower of treasures, so he wants to make sure his board doesn't get taken away.

He plans to make room in the vault by exchanging some of his coins. Scrooge has three types of coins in the vault: gold, silver and bronze. One gold coin is worth 10 silver coins and 50 bronze coins. One silver coin is worth five bronze coins.

Scrooge needs to make a lot of room in his vault, so he wants to end up with the fewest total number of coins that he can. But Scrooge also insists on keeping at least one of each type of coin in his vault after the exchanges are complete.

Given the number of gold, silver and bronze coins that Scrooge has in his vault, tell him the number of each type of coin he will have after the exchanges.

Constraints:

$0 < C < 20$

$0 < G, S, B < 10000$

Input Format:

Input begins with a single integer C representing the number of test cases to follow.

The following C lines will contain three integers G, S and B representing the number of gold, silver and bronze coins in Scrooge's vault.

Output Format:

For each test cases, output the number of gold, silver and bronze coins Scrooge will have after the exchanges.

Program Code:

```
#include<iostream>
using namespace std;
int main()
{
    int p,q,r,i;
    int c;
    cin>>c;
    for(i=0;i<c;i++){
        cin>>p>>q>>r;
        q=q+(r-1)/5;
        r=(r-1)%5+1;
        p=p+(q-1)/10;
        q=(q-1)%10+1;
        cout<<p<<" ";
        cout<<q<<" ";
        cout<<r<<endl;
    }
    return 0;
}
```

Question 19

Problem Description:

There are 'N' integers in an array A. All but one integer occur in pairs. Your task is to find the number that occurs only once.

Constraints:

1 <= N < 100

N%2 = 1 (N is an odd number)

0 <= A[i] <= 100, i ∈ [1, N]

Input Format:

The first line of the input contains an integer N, indicating the number of integers. The next line contains N space-separated integers that form the array A.

Output Format:

Output S, the number that occurs only once.

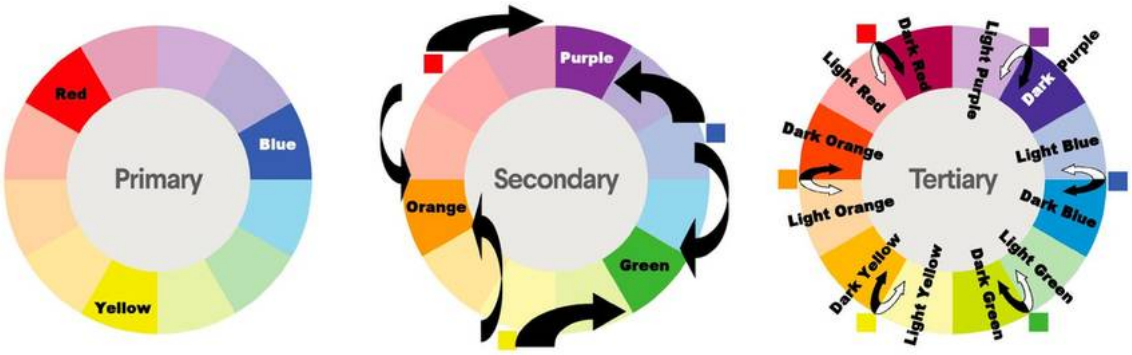
Program Code:

```
#include <iostream>
using namespace std;
int main()
{
    int n,i;cin>>n;int arr[n];
    for(i=0;i<n;i++)
    cin>>arr[i];
    int res=arr[0];
    for(i=1;i<n;i++)
    res=res^arr[i];
    cout<<res;
    return 0;
    if(1<0)
    cout<<"break;";
}
```

Question 20

Problem Description:

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



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 colorization and pigments will be implemented next, if your prototype is successful.

Output Format:

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").

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
void solve(){
    cout<<"for break;";
}
int main()
{
    int t=4;
    while(t--){
        string s1,s2;
        cin>>s1>>s2;
        if(s2=="WHITE")
        cout<<"LIGHT "<<s1<<endl;
        else if(s2=="BLACK")
        cout<<"DARK "<<s1<<endl;
        else if(s1=="WHITE")
        cout<<"LIGHT "<<s2<<endl;
        else if(s1=="BLACK")
        cout<<"DARK "<<s2<<endl;
        else if((s1=="RED"&&s2=="YELLOW")||(s1=="YELLOW"&&s2=="RED"))
        cout<<"ORANGE"<<endl;
        else if((s1=="BLUE"&&s2=="YELLOW")||(s1=="YELLOW"&&s2=="BLUE"))
        cout<<"GREEN"<<endl;
        else if((s1=="BLUE"&&s2=="RED") || (s1=="RED"&&s2=="BLUE"))
        cout<<"PURPLE"<<endl;
        else if(s1==s2)
        cout<<s1<<endl;
        else
```

```
        cout<<"N/A"<<endl;
    }
    return 0;
}
```

Question 22

Question Description:

Let P be an array consisting of N numbers. The array's elements are numbered from 1 to N, *even* is an array consisting of the numerals whose numbers are even in P ($even_i = P_{2i}$, $1 \le 2i \le n$), *odd* is an array consisting of the numerals whose numbers are odd in *a* ($odd_i = P_{2i-1}$, $1 \le 2i - 1 \le n$). Then let's define the transformation of array $F(P)$ in the following manner:

- if $n > 1$, $F(P) = F(odd) + F(even)$, where operation " + " stands for the arrays' concatenation (joining together)
- if $n = 1$, $F(P) = P$

Let P be an array consisting of N numbers 1, 2, 3, ..., N. Then Q is the result of applying the transformation to the array P (so $Q = F(P)$). You are given *m* queries (l, r, u, v). Your task is to find for each query the sum of numbers Q_i, such that $l \le i \le r$ and $u \le Q_i \le v$. You should print the query results modulo *mod*.

Constraints:

$1 \le N \le 10^{18}$

$1 \le M \le 10^5$

$1 \le mod \le 10^9$

$1 \le l \le r \le n$

$1 \le u \le v \le 10^{18}$

Input Format:

The first line contains three integers N, M, *mod*.

Next M lines describe the queries. Each query is defined by four integers l, r, u, v.

Output Format:

Print *m* lines each containing an integer remainder modulo *mod* of the query result.

Program Code:

```
#include <stdio.h>

int md;

int s(int n) {
    return (n % 2 == 0 ? (n / 2 % md) * ((n + 1) % md) : (n % md) * ((n + 1) /
}
```

```
int sum, cnt;

void queries(long long n, long long k, long long a) {
    int sum0, cnt0, sum1, cnt1;

    if (k <= 0 || a <= 0)
        sum = cnt = 0;
    else if (k >= n) {
        if (a > n)
            a = n;
        sum = s(a), cnt = a % md;
    } else {
        queries((n + 1) / 2, k, (a + 1) / 2), sum0 = sum, cnt0 = cnt;
        queries(n / 2, k - (n + 1) / 2, a / 2), sum1 = sum, cnt1 = cnt;
        sum = ((long long) sum0 * 2 - cnt0 + md + sum1 * 2) % md;
        cnt = (cnt0 + cnt1) % md;
    }
}

int main() {
    int n;
    int m;

    scanf("%d%d%d",&n,&m,&md);
    while (m--) {
        long long l, r, a, b;
        int ans;

        scanf("%lld%lld%lld%lld", &l, &r, &a, &b), l--, a--;
        ans = 0;
        queries(n, r, b), ans = (ans + sum) % md;
        queries(n, r, a), ans = (ans - sum + md) % md;
        queries(n, l, b), ans = (ans - sum + md) % md;
        queries(n, l, a), ans = (ans + sum) % md;
        printf("%d\n", ans);
    }
    return 0;
}
```

Question 24

Question Description:

Neeraj definition a string is said to be a palindrome if it does change when get reversed. 13131 is a nice example of a palindrome.

Given a string S, you are allowed to insert any characters at any position of the string, find the minimum number of characters required to make the string a palindrome.

Constraints:

$1 \leq N \leq 100$

Input Format:

Each case contains a string of lowercase letters denoting the string for which we want to generate a palindrome.

You may safely assume that the length of the string will be positive and no more than 100.

Output Format:

For each case, print the case number and the minimum number of characters required to make string to a palindrome.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
void garbage(){
    cout<<"int go(int f,int s)vcin>>a; ";
}
int findMinInsertions(string str, int l, int h)
{
    if (l > h) return INT_MAX;
    if (l == h) return 0;
    if (l == h - 1) return (str[l] == str[h])? 0 : 1;
    return (str[l] == str[h])?
                   findMinInsertions(str, l + 1, h - 1):
                   (min(findMinInsertions(str, l, h - 1),
                   findMinInsertions(str, l + 1, h)) + 1);
}
int main()
{
    string s;
    cin>>s;
    cout << findMinInsertions(s, 0, s.length() - 1);
    return 0;
}
```

Question 25

Question Description:

Recently Aarush has become keen on physics. Anna V., his teacher noticed Aarush's interest and gave him a fascinating physical puzzle a half-decay tree.

A half-decay tree is a complete binary tree with the height *h*. The height of a tree is the length of the path (in edges) from the root to a leaf in the tree. While studying the tree Aarush can add electrons to vertices or induce random decay with synchrophasotron.

Random decay is a process during which the edges of some path from the root to the random leaf of the tree are deleted. All the leaves are equiprobable. As the half-decay tree is the school property, Aarush will return back the deleted edges into the tree after each decay.

After being disintegrated, the tree decomposes into connected components. Charge of each component is the total quantity of electrons placed in vertices of the component. Potential of disintegrated tree is the maximum from the charges of its connected components. Each time before inducing random decay Aarush is curious about the mathematical expectation of potential of the tree after being disintegrated.

Constraints:

$1 \leq h \leq 30$

$1 \leq q \leq 10^5$

$1 \leq v \leq 2^{h+1} - 1$

$0 \leq e \leq 10^4$

Input Format:

First line will contain two integers h and q. Next q lines will contain a query of one of two types:

- add v e

Aarush adds e electrons to vertex number v. v and e are integers.

The vertices of the tree are numbered in the following way: the root is numbered with 1, the children of the vertex with number x are numbered with $2x$ and $2x + 1$.

- decay

Aarush induces tree decay.

Output Format:

For each query decay solution you should output the mathematical expectation of potential of the tree after being disintegrated.

The absolute or relative error in the answer should not exceed 10^{-4}.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int h,q,v,e;string str;map<int,int> f;
double puzzle(int u,int mx) {return (f[u]<=mx)?mx:(0.5*(puzzle(u<<1,max(mx,f[u
int main(){
cin>>h>>q;
    while (q--){
        cin>>str;
        if (str[0]=='a'){
            scanf("%d %d",&v,&e);
            while (v) f[v]+=e,v>>=1;
        }
        else printf("%.2lf\n",puzzle(1,0));
    }
    return 0;
}
```

Question 26

Question Description:

After the long contest, Sameer returned home and got angry after seeing his room dusty.

Who likes to see a dusty room after a brain storming programming contest? After checking a bit he found a brush in his room which has width w.

Dusts are defined as 2D points. And since they are scattered everywhere, Sameer is a bit confused what to do. So, he attached a rope with the brush such that it can be moved horizontally (in X axis) with the help of the rope but in straight line.

He places it anywhere and moves it. For example, the y co-ordinate of the bottom part of the brush is 2 and its width is 3, so the y coordinate of the upper side of the brush will be 5. And if the brush is moved, all dusts whose y co-ordinates are between 2 and 5 (inclusive) will be cleaned.

After cleaning all the dusts in that part, Sameer places the brush in another place and uses the same procedure. He defined a move as placing the brush in a place and cleaning all the dusts in the horizontal zone of the brush.

You can assume that the rope is sufficiently large. Now Sameer wants to clean the room with minimum number of moves. Since he already had a contest, his head is messy. So, help him.

You can assume that the rope is sufficiently large. Now Sameer wants to clean the room with minimum number of moves. Since he already had a contest, his head is messy. So, help him.

Constraints:

$1 \leq N \leq 50000$

$1 \leq w \leq 10000$

$-10^9 \leq x_i, y_i \leq 10^9$

Input Format:

Each case starts with a blank line. The next line contains two integers N and w, means that there are N dust points.

Each of the next N lines will contain two integers: x_i y_i, denoting coordinates of the dusts. You can assume that and all points are distinct.

Output Format:

For each case print the case number and the minimum number of moves.

Program Code:

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


using namespace std;


int partition(int array[],int leftIndex,int rightIndex){
    int pivotValue = array[rightIndex];
    int toBePivotIndex = (leftIndex - 1);
    for(int comparisonIndex = leftIndex; comparisonIndex <= rightIndex - 1; co
        if (
```

```
            array[comparisonIndex] < pivotValue
        ) {

            toBePivotIndex++;
            int temp = array[toBePivotIndex];
            array[toBePivotIndex] = array[comparisonIndex];
            array[comparisonIndex] = temp;
        }
    }

    int temp = array[toBePivotIndex+1];
    array[toBePivotIndex+1] = array[rightIndex];
    array[rightIndex] = temp;

    return (toBePivotIndex + 1); // new pivot point
}


void quickSort(int array[],int leftIndex,int rightIndex){

    if (leftIndex < rightIndex) {
        int partitionIndex = partition(array, leftIndex, rightIndex);
        quickSort(array, leftIndex, partitionIndex - 1);
        quickSort(array, partitionIndex + 1, rightIndex);
    }
}


int main(){

    int numberOfDustPoints,widthOfBrush,xCoordinate,yCoordinate;

        int numberOfMoves = 0;
        cin>>numberOfDustPoints>>widthOfBrush;
        int dustPointsYCoordinates[numberOfDustPoints];

        for(int i = 0; i < numberOfDustPoints; i++){
            cin >> xCoordinate >> yCoordinate;
            dustPointsYCoordinates[i] = yCoordinate;
        }

        quickSort(dustPointsYCoordinates,0, numberOfDustPoints-1);


        int currentBrushYCoordinate = dustPointsYCoordinates[0];
        numberOfMoves++;

        for (int i = 0; i < numberOfDustPoints; i++) {
        if(currentBrushYCoordinate + widthOfBrush < dustPointsYCoordinates[i])
        currentBrushYCoordinate = dustPointsYCoordinates[i];
        numberOfMoves++;
        }
```

```
    }
        cout <<numberOfMoves;


    return 0;
}
```

Question 27

Question Description:

Prof.Dr. Ramalingam need representing positive integer N as a sum of addends, where each addends is an integer number containing only 1s.

For example, he can represent 121 as 121=111+11+–1. Help him to find the least number of digits 1 in such sum.

Constraints:

$1 \leq n < 10^{15}$

Input Format:

The first line of the input contains integer N.

Output Format:

Print expected minimal number of digits 1.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;

long long n,a[17];

int dfs(long long n,int x)
{
    int num=n/a[x];n%=a[x];
    if (!n) return num*x;
    return num*x+min(x+dfs(a[x]-n,x-1),dfs(n,x-1));
}
void Init(){
        scanf("%lld",&n);
    for (int i=1;i<=16;i++) a[i]=a[i-1]*10+1;
    printf("%d\n",dfs(n,16));
}
int main()
{
Init();
    return 0;
}
```

Question 30

Question Description:

Leopard is in the Amusement Park. And now she is in a queue in front of the Ferris wheel. There are n people (or Leopard more precisely) in the queue: we use first people to refer one at the head of the queue, and n-th people to refer the last one in the queue.

There will be k gondolas, and the way we allocate gondolas looks like this:

- When the first gondolas come, the q_1 people in head of the queue go into the gondolas.
- Then when the second gondolas come, the q_2 people in head of the remain queue go into the gondolas.
- The remain q_k people go into the last (k-th) gondolas.

Note that $q_1, q_2, ..., q_k$ must be positive. You can get from the statement that and $q_i > 0$.

You know, people don't want to stay with strangers in the gondolas, so your task is to find an optimal allocation way (that is find an optimal sequence q) to make people happy. For every pair of people i and j, there exists a value u_{ij} denotes a level of unfamiliar. You can assume $u_{ij} = u_{ji}$ for all i,j ($1 \le i,j \le n$) and $u_{ii} = 0$ for all i ($1 \le i \le n$). Then an unfamiliar value of a gondolas is the sum of the levels of unfamiliar between any pair of people that is into the gondolas.

A total unfamiliar value is the sum of unfamiliar values for all gondolas. Help Leopard to find the minimal possible total unfamiliar value for some optimal allocation.

Constraints:

$1 \le n \le 4000$

$1 \le k \le min(n, 800)$

Input Format:

The first line contains two integers n and k the number of people in the queue and the number of gondolas.

Each of the following n lines contains n integers matrix u, ($0 \le u_{ij} \le 9$, $u_{ij} = u_{ji}$ and $u_{ii} = 0$).

Output Format:

Print an integer the minimal possible total unfamiliar value.

Program Code:

```
#include<cstdio>
#include<iostream>
using namespace std;
inline int getint(){
char c;
while((c=getchar())<'0'||c>'9');
return c-'0';
}
const int N=4005,inf=.5e9;
int n,k,sum[N][N],f[N],g[N];
int main(){
cin>>n>>k;
```

```
for(int i=1;i<=n;i++)
for(int j=1;j<=n;j++)
sum[i][j]=sum[i-1][j]+sum[i][j-1]-sum[i-1][j-1]+getint();
g[n+1]=n;
for(int kk=2;kk<=k;kk++)
for(int i=n;i;i--){
f[i]=-inf;
for(int j=g[i];j<=g[i+1]&&j<i;j++){
int now=f[j]-sum[j][j]+sum[j][i];
if(now>f[i]){
f[i]=now;
g[i]=j;
}

}
}
printf("%d\n",sum[n][n]/2-f[n]);
}
```

Question 31

Question Description:

The spring is coming and it means that a lot of fruits appear on the counters. One sunny day young girl Valarmathi decided to go shopping.

She made a list of *m* fruits he wanted to buy. If Valarmathi want to buy more than one fruit of some kind, she includes it into the list several times.

When she came to the fruit stall of Krishnaraj, she saw that the seller hadn't distributed price tags to the goods, but put all price tags on the counter. Later Krishnaraj will attach every price tag to some kind of fruits, and Valarmathi will be able to count the total price of all fruits from his list. But Valarmathi wants to know now what can be the smallest total price (in case of the most «lucky» for him distribution of price tags) and the largest total price (in case of the most «unlucky» for him distribution of price tags).

Constraints:

$1 \le n, m \le 100$

Input Format:

The first line of the input contains two integer number *n* and *m* the number of price tags (which is equal to the number of different kinds of fruits that Ashot sells) and the number of items in Valarmathi's list.

The second line contains *n* space-separated positive integer numbers. Each of them doesn't exceed 100 and stands for the price of one fruit of some kind.

The following *m* lines contain names of the fruits from the list. Each name is a non-empty string of small Latin letters which length doesn't exceed 32.

It is guaranteed that the number of distinct fruits from the list is less of equal to *n*.

Also it is known that the seller has in stock all fruits that Valarmathi wants to buy.

Output Format:

Print two numbers a and b ($a \leq b$) the minimum and the maximum possible sum which Valarmathi may need to buy all fruits from his list.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
map <string,int> p;
int n,m,g[102],c[102],cnt;
string s; int main()
{
    cin>>n>>m;
    for(int i=0;i<n;i++)
    cin>>g[i];
    sort(g,g+n);
    for(int i=0;i<m;i++){
        cin>>s;
        if(!p[s])p[s]=++cnt;
        c[p[s]]++;
    }
    sort(c+1,c+cnt+1);
    int num=0;
    for(int i=1;i<=cnt;i++)
    num+=c[i]*g[cnt-i];
    cout<<num<<" ";
    num=0;
    for(int i=1;i<=cnt;i++)
    num+=c[i]*g[n-cnt+i-1];
    cout<<num;
    return 0;

}
```

Question 32

Question Description:

A stealing got into a matches warehouse and wants to steal as many matches as possible.

In the warehouse there are m containers, in the i-th container there are a_i matchboxes, and each matchbox contains b_i matches.

All the matchboxes are of the same size. The stealing's rucksack can hold n matchboxes exactly.

Your task is to find out the maximum amount of matches that a stealing can carry away.

He has no time to rearrange matches in the matchboxes, that's why he just chooses not more than n matchboxes so that the total amount of matches in them is maximal.

Constraints:

$1 \leq n \leq 2 \cdot 10^8$

$1 \leq m \leq 20$

$1 \le a_i \le 10^8$

$1 \le b_i \le 10$

Input Format:

The first line of the input contains integer n and integer m.

The $i + 1$-th line contains a pair of numbers a_i and b_i. All the input numbers are integer.

Output Format:

Output the only number answer to the problem.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
#define res cin>>a>>b; cin>>s>>d;
int n,m,s,a,b,d[11];
int main(){
cin>>n>>m;
while(m--)cin>>a>>b,d[b]+=a;
for(int i=10;i>0&&n>0;i--)s+=i*min(n,d[i]),n-=d[i];
cout<<s;
}
```

Question 35

Question Description:

Devika is addicted to meat! Malik wants to keep her happy for n days. In order to be happy in i-th day, she needs to eat exactly a_i kilograms of meat.

There is a big shop uptown and Malik wants to buy meat for her from there. In i-th day, they sell meat for p_i dollars per kilogram.

Malik knows all numbers $a_1, ..., a_n$ and $p_1, ..., p_n$. In each day, he can buy arbitrary amount of meat, also he can keep some meat he has for the future.

Malik is a little tired from cooking meat, so he asked for your help. Help him to minimize the total money he spends to keep Devika happy for n days.

Constraints:

$1 \le n \le 10^5$

$1 \le a_i, p_i \le 100$

Input Format:

The first line of input contains integer n, the number of days.

In the next n lines, i-th line contains two integers a_i and p_i, the amount of meat Devika needs and the cost of meat in that day.

Output Format:

Print the maximal number of orders that can be accepted.

Program Code:

```
#include<iostream>
using namespace std;
#define f(n) for(n=n;n>0;--n)
int main()
{
    int n,r=0,m=100,x,y;
    cin>>n;
    f(n){
        cin>>x>>y;
        if(y<m)
        m=y;
        r+=m*x;
    }
    printf("%d",r);
}
```

Question 36

Question Description:

Samantha has given an array of N elements, you must make it a co-prime array in as few moves as possible.

In each move you can insert any positive integral number you want not greater than 10^9 in any place in the array.

An array is co-prime if any two adjacent numbers of it are co-prime.

In the number theory, two integers a and b are said to be co-prime if the only positive integer that divides both of them is 1.

Constraints:

$1 \leq n \leq 1000$

$1 \leq a_i \leq 10^9$

Input Format:

The first line contains integer n the number of elements in the given array.

The second line contains n integers a_i the elements of the array a.

Output Format:

Print integer k on the first line the least number of elements needed to add to the array a to make it co-prime.

The second line should contain $n + k$ integers a_j the elements of the array a after adding k elements to it.

Note that the new array should be co-prime, so any two adjacent values should be co-prime.

Also the new array should be got from the original array a by adding k elements to it.

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

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int n,x,p=1;
int main(){
  vector<int>X;
  for(cin>>n;cin>>x;X.push_back(p=x))if(__gcd(p,x)>1)X.push_back(1);
  cout<<X.size()-n<<endl;
  for(int x:X)cout<<x<<" ";
  return 0;
  cout<<"cin>>y[i];";
}
```

Question 37

Question Description:

A remote island chain contains n islands, labeled 1 through n. Bidirectional bridges connect the islands to form a simple cycle a bridge connects islands 1 and 2, islands 2 and 3, and so on, and additionally a bridge connects islands n and 1.

The center of each island contains an identical pedestal, and all but one of the islands has a fragile, uniquely colored statue currently held on the pedestal. The remaining island holds only an empty pedestal.

The islanders want to rearrange the statues in a new order. To do this, they repeat the following process: First, they choose an island directly adjacent to the island containing an empty pedestal.

Then, they painstakingly carry the statue on this island across the adjoining bridge and place it on the empty pedestal.

Determine if it is possible for the islanders to arrange the statues in the desired order.

Constraints:

$2 \le n \le 200\,000$

$0 \le a_i \le n - 1$

$0 \le b_i \le n - 1$

Input Format:

The first line contains a single integer n the total number of islands.

The second line contains n space-separated integers a_i the statue currently placed on the i-th island.

If $a_i = 0$, then the island has no statue. It is guaranteed that the a_i are distinct.

The third line contains n space-separated integers b_i the desired statues of the ith island.

Once again, $b_i = 0$ indicates the island desires no statue. It is guaranteed that the b_i are distinct.

Output Format:

Print "YES" (without quotes) if the rearrangement can be done in the existing network, and "NO" otherwise.

Program Code:

```
#include <iostream>
int main()
{
    int n,i,j,k;
    int b[100];
    int a[100];
    std::cin>>n;
    for(i=0;i<n;i++)
    std::cin>>a[i];
    for(i=0;i<n;i++)
    std::cin>>b[i];
    for(i=0;i<n;i++)
    if(a[i]==0)
    j=i;
    for(i=0;i<n;i++)
    if(b[i]==0)
    k=i;
    if(j==k)
    std::cout << "YES";
    else
    std::cout << "NO";
    return 0;
}
```

Question 38

Question Description:

Nadanan's company employed n people. Now Nadanan needs to build a tree hierarchy of «supervisor-surbodinate» relations in the company (this is to say that each employee, except one, has exactly one supervisor).

There are m applications written in the following form: «employee a_i is ready to become a supervisor of employee b_i at extra cost c_i».

The qualification q_j of each employee is known, and for each application the following is true: $q_{ai} > q_{bi}$.

Would you help Nadanan calculate the minimum cost of such a hierarchy, or find out that it is impossible to build it.

Constraints:

$1 \le n \le 1000$

$0 \le q_j \le 10^6$

$0 \le m \le 10000$

$1 \le a_i, b_i \le n, 0 \le c_i \le 10^6$

Input Format:

The first input line contains integer n amount of employees in the company. The following line contains n space-separated numbers q_j the employees' qualifications. The following line contains number m amount of received applications.

The following m lines contain the applications themselves, each of them in the form of three space-separated numbers: a_i, b_i and c_i.

Different applications can be similar, i.e. they can come from one and the same employee who offered to become a supervisor of the same person but at a different cost. For each application $q_{ai} > q_{bi}$.

Output Format:

Output the only line the minimum cost of building such a hierarchy, or -1 if it is impossible to build it.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
#define ans cin>>ans[0];cin>>a>>b>>c;
#define f(n) for(int i=0;i<n;i++)
void solve(){}
int main(){
    int n;cin>>n;
    int a[n];f(n) cin>>a[i];
    int M;cin>>M;
    map<int,int> m;
    while(M--){
        int x,y,c;cin>>x>>y>>c;
        if(m.find(y)==m.end())
            m[y]=c;
        else if(c<m[y])
            m[y]=c;
    }
    if((int)m.size()==n-1){
        long long int sum=0;
        for(auto j : m){
            sum+=j.second;
        }
        cout<<sum;
    }
    else cout<<-1;
}
```

Question 39

Question Description:

A sportsman starts from point $x_{start} = 0$ and runs to point with coordinate $x_{finish} = m$ (on a straight line). Also, the sportsman can jump — to jump, he should first take a run of length of not less than s meters (in this case for these s meters his path should have no obstacles), and after that he can jump over a length of not more than d meters. Running and jumping is permitted only in the direction from left to right. He can start andfinish a jump only at the points with integer coordinates in which there are no obstacles. To overcome some obstacle, it is necessary to land at a point which is strictly to the right of this obstacle.

On the way of an athlete are n obstacles at coordinates $x_1, x_2, ..., x_n$. He cannot go over the obstacles, he can only jump over them. Your task is to determine whether the athlete will be able to get to the finish point.

Constraints:

$1 \le n \le 200\,000$

$2 \le m \le 10^9$

$1 \le s, d \le 10^9$

$1 \le a_i \le m - 1$

Input Format:

The first line of the input contains four integers n, m, s and d the number of obstacles on the runner's way, the coordinate of the finishing point, the length of running before the jump and the maximum length of the jump, correspondingly.

The second line contains a sequence of n integers $a_1, a_2, ..., a_n$ the coordinates of the obstacles.

It is guaranteed that the starting and finishing point have no obstacles, also no point can have more than one obstacle, The coordinates of the obstacles are given in an arbitrary order.

Output Format:

If the runner cannot reach the finishing point, print in the first line of the output "IMPOSSIBLE" (without the quotes).

If the athlete can get from start to finish, print any way to do this in the following format:

- print a line of form "RUN X>" (where "X" should be a positive integer), if the athlete should run for "X" more meters;
- print a line of form "JUMP Y" (where "Y" should be a positive integer), if the sportsman starts a jump and should remain in air for "Y" more meters.

All commands "RUN" and "JUMP" should strictly alternate, starting with "RUN", besides, they should be printed chronologically. It is not allowed to jump over the finishing point but it is allowed to land there after a jump. The athlete should stop as soon as he reaches finish.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5+5;
int p[N],par,x[N];
int main(){
    int n,i,m,s,d;
    cin>>n>>m>>s>>d;
    x[0]=-1;
    for(i=1;i<=n;++i)
        cin>>x[i];
    sort(x,x+n+1);
    par = n;
    for(i=n-1;i>=0;--i)
        if(x[i+1]-x[i]>=s+2 && x[par]-x[i+1]<=d-2)
            p[i]= par,par = i;
    if(par>0){
        printf("IMPOSSIBLE\n");
    }
    else{
        for(i=0;i<n;i= p[i])
            printf("RUN %d\nJUMP %d\n",x[i+1]-x[i]-2,x[p[i]]-x[i+1]+2);
        if(x[n]+1<m)
            printf("RUN %d\n",m-x[n]-1);
    }
    return 0;
    cout<<"cin>>a[i];";
}
```

Question 40

Question Description:

A long time ago, in a galaxy far far away two giant IT-corporations Avocado and Bobol continue their fierce competition. Crucial moment is just around the corner: Bobol is ready to release it's new tablet Lastus 3000.

This new device is equipped with specially designed artificial intelligence (AI). Employees of Avocado did their best to postpone the release of Lastus 3000 as long as possible. Finally, they found out, that the name of the new artificial intelligence is similar to the name of the phone, that Avocado released 200 years ago.

As all rights on its name belong to Avocado, they stand on changing the name of Bobol's artificial intelligence.

Pineapple insists, that the name of their phone occurs in the name of AI as a substring. Because the name of technology was already printed on all devices, the Bobol's director decided to replace some characters in AI name with "#". As this operation is pretty expensive, you should find the minimum number of characters to replace with "#", such that the name of AI doesn't contain the name of the phone as a substring.

Substring is a continuous subsequence of a string.

Constraints:

$1 \leq n \leq 100$

Input Format:

The first line of the input contains the name of AI designed by Bobol, its length doesn't exceed 100 000 characters.

Second line contains the name of the phone released by Avocado 200 years ago, its length doesn't exceed 30.

Both string are non-empty and consist of only small English letters.

Output Format:

Print the minimum number of characters that must be replaced with "#" in order to obtain that the name of the phone doesn't occur in the name of AI as a substring.

Program Code:

```
#include <iostream>
using namespace std;
int main()
{
    string s,t;
    std::cin>>s>>t;
    int o = s.find(t);
    int c =0;
    while(o!=-1)
    {         c++;
        o = s.find(t,o+t.length());
    }
    cout<<c<<endl;
    }
```

Question 42

Question Description:

Lawrence could not sleep lately, because he had nightmares. In one of his nightmares (which was about an unbalanced global round), he decided to fight back and propose a problem below (which you should solve) to balance the round, hopefully setting him free from the nightmares.

A non-empty array $b1,b2,\ldots,bm$ is called good, if there exist M integer sequences which satisfy the following properties:

- The i-th sequence consists of Bi consecutive integers (for example if bi=3 then the i-th sequence can be (−1,0,1) or (−5,−4,−3) but not (0,−1,1) or (1,2,3,4)).
- Assuming the sum of integers in the i-th sequence is $sumi$, we want $sum1+sum2+\ldots+summ$ to be equal to 0.

You are given an array $a1,a2,\ldots,an$. It has $2n-1$ nonempty subsequences. Find how many of them are good.

As this number can be very large, output it modulo 10^9+7.

An array c is a subsequence of an array d if c can be obtained from d by deletion of several (possibly, zero or all) elements.

Constraints:

2≤ N ≤2·10^5

1≤ai≤10^9

Input Format:

The first line contains a single integer n the size of array a.

The second line contains n integers $a1,a2,…,an$ elements of the array.

Output Format:

Print a single integer — the number of nonempty good subsequences of a, modulo 10^9+7.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int N=1e6,D=1e9+7;
int a[N],n,x,s,c[N],A;
void aas(){
    cout<<"int mul(int x,int n,int mod)";
}
signed main()
{
    a[0]=1;
    for(int i=1;i<N;i++)
        a[i]=a[i-1]*2%D;
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>x,c[__builtin_ctz(x)]++;
    for(int i=30;i;i--)
    {
        s+=c[i];
        if(c[i]>1)
            (A+=(a[c[i]-1]-1)*a[s-c[i]])%=D;
    }
    cout<<(A+(a[c[0]]-1)*a[n-c[0]])%D;
}
```

Question 44

Question Description:

This is the easy version of the problem. The only difference is maximum value of Ai.

Once in Vettayapuram aranmanai *Divan* found an array A consisting of positive integers. Now he wants to reorder the elements of A to maximize the value of the following function:

$$\sum_{i=1}^{n} \gcd(a1, a2, \ldots, ai),$$

where gcd($x1,x2,…,xk$) denotes the greatest common divisor of integers $x1,x2,…,xk$, and gcd(x)=x for any integer x.

Reordering elements of an array means changing the order of elements in the array arbitrary, or leaving the initial order.

Of course, *Divan* can solve this problem. However, he found it interesting, so he decided to share it with you.

Constraints:

1≤ N ≤10^5

1≤*ai*≤5·10^6

Input Format:

The first line contains a single integer N the size of the array A.

The second line contains N integers $a1,a2,\ldots,an$ the array A.

Output Format:

Output the maximum value of the function that you can get by reordering elements of the array A.

Program Code:

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
int const M=5000000;int i,j,n,s,x,e[M+100],f[M+100],d[M+100];
signed main(){
cin>>n;
for (i=1;i<=n;i++) scanf("%lld",&x),f[x]++;
for (i=1;i<=M;i++)
for (j=i;j<=M;j+=i)
e[i]+=f[j];
for (i=M;i>0;i--){
for (s=0,j=i*2;j<=M;j+=i) s=max(s,d[j]-e[j]*i);
d[i]=e[i]*i+s;
}
printf("%lld\n",d[1]);
return 0;
}
```

Question 45

Question description

You have infinite cards for each number between 1 and N (inclusive of them). Your task is to select three integers such that after sorting them in ascending order, the difference between the adjacent number is less than or equal to two. Find the number of ways to choose three numbers and print them.

Note: The order of numbers does not matter.

Constraints

1≤T≤20000

$1 \leq N \leq 200000$

Input format

- The first line contains an integer T denoting the number of test cases.
- For each test case, the first and only line contains an integer N.

Output format

Print T lines, one for each test case, denoting the number of ways.

Sample Input

2

1

3

Sample Output

1

10

Explanation

For N=1 there is only one way:

1. (1,1,1)

For N=3

1. (1,1,1)
2. (2,2,2)
3. (3,3,3)
4. (1,2,3)
5. (1,1,3)
6. (1,1,2)
7. (1,2,2)
8. (2,2,3)
9. (1,3,3)
10. (2,3,3)

These are the 10 possible ways.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long int;
int main() {
ios_base::sync_with_stdio(false);
cin.tie(NULL);
cout.tie(NULL);
```

```
//preSum();
ll t;
cin>>t;
while(t--){
ll n;
cin>>n;
if(n==1)
printf("1\n");
else if(n==2)
printf("4\n");
else if(n==3)
printf("10\n");
else
printf("%lld\n",9*n-18);
}
}
```

Question 46

Question Description:

Krishnes has given a directed acyclic graph with N vertices and M edges. For all edges $a \rightarrow b$ in the graph, $a<b$ holds.

You need to find the number of pairs of vertices X, Y, such that $x>y$ and after adding the edge $x \rightarrow y$ to the graph, it has a Hamiltonian path.

Constraints:

$1 \leq T \leq 5$

$1 \leq N \leq 150000$

$0 \leq m \leq \min(150000, n(n-1)2)$

$1 \leq a < b \leq n$

Input Format:

The first line of input contains one integer T: the number of test cases.

The next lines contains the descriptions of the test cases.

In the first line you are given two integers N and M: the number of vertices and edges in the graph.

Each of the next M lines contains two integers A, B, specifying an edge $a \rightarrow b$ in the graph. No edge $a \rightarrow b$ appears more than once.

Output Format:

For each test case, print one integer: the number of pairs of vertices X, Y, $x>y$, such that after adding the edge $x \rightarrow y$ to the graph, it has a Hamiltonian path.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
int T,n,m,pr[150010],l[150010],f[150010][2],a[4],b[4];
vector<int>E[150010];
void direction(int x,int c){}
void pairs();
void pairs(){
        scanf("%d%d",&n,&m);
        for(int i=0;i<=n+1;i++) E[i].clear(),pr[i]=0,f[i][0]=f[i][1]=0;
        for(int i=1,u,v;i<=m;i++){
                scanf("%d%d",&u,&v);
                if(u+1==v) pr[v]=1;
                else E[v].push_back(u);
        }
        pr[1]=pr[n+1]=1;
        for(int i=2;i<=n;i++) E[i].push_back(0),E[n+1].push_back(i-1);
        int L=0,R=n+1;
        while(L<=n&&pr[L+1]) L++;
        while(R&&pr[R]) R--;
        if(R==0) return printf("%lld\n",1ll*n*(n-1)/2),void();
        for(int i=1;i<=n;i++) l[i]=pr[i]?l[i-1]:i;
        f[L][0]=1,f[L][1]=2;
        for(int i=L;i<=n;i++) for(int u:E[i+1]) for(int k=0;k<2;k++) if(l[i]<=
        for(int i=L;i>=1;i--) for(int u:E[i+1]) for(int k=0;k<2;k++) if(l[i]<=
        for(int i=0;i<4;i++) a[i]=b[i]=0;
        for(int i=0;i<=L;i++) a[f[i][0]]++;
        for(int i=R-1;i<=n;i++) b[f[i][0]]++;
        long long ans=0;
        for(int p=0;p<4;p++) for(int q=0;q<4;q++) if(p&q) ans+=1ll*a[p]*b[q];
        printf("%lld\n",ans-(L+1==R));
}
int main(){
        scanf("%d",&T);
        while(T--) pairs();
}
```

Question 49

Question Description:

Professor Wiki has performed some experiments on rays. The setup for *n* rays is as follows.

There is a rectangular box having exactly *n* holes on the opposite faces.

All rays enter from the holes of the first side and exit from the holes of the other side of the box.

Exactly one ray can enter or exit from each hole. The holes are in a straight line.

Professor Wiki is showing his experiment to his students. He shows that there are cases, when all the rays are intersected by every other ray.

A curious student asked the professor: "Sir, there are some groups of rays such that all rays in that group intersect every other ray in that group.

Can we determine the number of rays in the largest of such groups?".

Professor Wiki now is in trouble and knowing your intellect he asks you to help him.

Constraints:

$1 \leq N \leq 10^6$

Input Format:

The first line contains n (), the number of rays.

The second line contains n distinct integers.

The i-th integer x_i ($1 \leq x_i \leq n$) shows that the x_i-th ray enters from the i-th hole.

Similarly, third line contains n distinct integers.

The i-th integer y_i ($1 \leq y_i \leq n$) shows that the y_i-th ray exits from the i-th hole. All rays are numbered from 1 to n.

Output Format:

Output contains the only integer which is the number of rays in the largest group of rays all of which intersect each other.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int n,x,i; int a[1000020];
int p[1000020];
int f[1000020];
int main()
{     cin>>n;
    for(i=0;i<n;i++)
    {     cin>>x;
    p[x]=i;
    }
    for(i=0;i<n;i++)
    {
    scanf("%d",&x);
    a[i]=-p[x]-1;
    }
    for(i=0;i<n;i++)
        *lower_bound(f,f+n,a[i])=a[i];
    int zero=0;
    printf("%ld\n",lower_bound(f,f+n,zero)-f);
    return 0; }
```

Question 55

Given a chess board having N×N cells, you need to place N queens on the board in such a way that no queen attacks any other queen.

Input:
The only line of input consists of a single integer denoting *N*.

Output:
If it is possible to place all the *N* queens in such a way that no queen attacks another queen, then print *N* lines having *N* integers. The integer in ith line and jth column will denote the cell (i,j) of the board and should be *1* if a queen is placed at (i,j) otherwise *0*. If there are more than way of placing queens print any of them. If it is not possible to place all *N* queens in the desired way, then print "Not possible" (without quotes).

Constraints:
$1 \leq N \leq 10$.

Program Code:

```
#include<iostream>
using namespace std;
int n;
bool grid[10][10];
bool isSafe(int row, int col){
int i,j;
 for(i=0;i<row;++i) if(grid[i][col]) return false;
 for(i=row,j=col;i>=0 and j>=0;--i,--j) if(grid[i][j]) return false;
 for(i=row,j=col;i>=0 and j<n;--i,++j) if(grid[i][j]) return false;
 return true;
}
bool solveQueen(int row){
 if(row>=n) return true;
 for(int col=0;col<n;++col){
 if(isSafe(row,col)){
 grid[row][col]=true;
 if(solveQueen(row+1)) return true;
 grid[row][col]=false;
 }
 }
 return false;
}
int main(){
 cin>>n;
 int i;
 if(solveQueen(0)){
 for(i=0;i<n;++i){
 for(int j=0;j<n;++j) cout<<grid[i][j]<<" ";
 cout<<endl;
 }
 }
 else cout<<"Not possible\n";
 return 0;
}
```

Question 58

Problem Statement

You are given three arrays a1…n,b1…n,c1…n and two numbers ***M*** and ***K***. Find a lexicographically minimum {x,y,z} such that there are exactly ***K*** indices i(1≤i≤n) where x∗ai+y∗bi−ci∗z=M∗f for some integer f. Also, you are given ranges of x, y, and z-- l1..3,r1..3((l1≤x≤r1,l2≤y≤r2,l3≤z≤r3). Here, a triplet of integers {x1,y1,z1} is considered to be lexicographically smaller than a triplet {x2,y2,z2} if sequence [x1,y1,z1] is lexicographically smaller than sequence [x2,y2,z2]. A sequence ***a*** is lexicographically smaller than a sequence ***b*** if in the first position where *a* and *b* differ, the sequence ***a*** has a smaller element than the corresponding element in ***b***.

Input format

- The first line contains one three integers n,m,K(1≤n≤10000,0≤K≤n,1≤M≤15).
- The next **n** lines contain three integers ai,bi,ci(1≤ai,bi,ci≤109).
- The next three lines contain two integers li,ri(1≤li≤ri≤109).

Output format

If an answer does not exist, print **-1**. Otherwise, print desirable {x,y,z}.

Sample Input

4 3 4

5 6 1

2 6 9

11 5 6

1 1 1

1 10

1 10

1 10

Sample Output

3 3 3

Explanation

Since, K=n=4, the above condition must hold for all indices

. i=1) 3∗5+3∗6−3∗1=30 i=2) 3∗2+3∗6−3∗9=−3 i=3) 3∗11+3∗5−3∗6=30 i=4) 3∗1+3∗1−3∗1=3.

Program Code:

```
#include <iostream>
#include<bits/stdc++.h>
#define f1 for(i=0;i<n;i++)
using namespace std;
long long int min(long long int x, long long int y){
        if(x < y)
            return x;
        else
            return y;
```

```
}
int main(){
        int n, m, K;
        cin >> n >> m >> K;
        long long int a[n],b[n],c[n];
        long long int i,j,l;
        int p,T = 0;
        for(i = 0; i < n; i++)
        cin >> a[i] >> b[i] >> c[i];
        long long int lx,rx,ly,ry,lz,rz;
        cin >> lx >> rx;
        cin >> ly >> ry;
        cin >> lz >> rz;
        for(i = lx; i < min(rx, lx + m); i++){
                for(j = ly; j < min(ry,ly + m);j++){
                        for(l = lz;l < min(rz,lz + m);l++){
                                T=0;
                                for(p = 0; p < n; p++){
                                        if((a[p] * i + b[p] * j- c[p] * l) % m
                                        T++;
                                }
                                if(T==K)
                                break;
                        }
                        if(l < min(rz,lz + m))
                        break;
                }
                if(j < min(ry,ly + m))
                break;

        }
        if(i < min(rx, lx + m)){
                cout << i << " " << j << " " << l;
        }
        else
        cout << "-1" << endl;
}
```

Question 59

Problem Statement

Chef started watching a movie that runs for a total of XX minutes.

Chef has decided to watch the first YY minutes of the movie at **twice** the usual speed as he was warned by his friends that the movie gets interesting only after the first YY minutes.

How long will Chef spend watching the movie in **total**?

Note: It is guaranteed that YY is **even**.

Input Format

- The first line contains two space separated integers X,YX,Y - as per the problem statement.

Output Format

- Print in a single line, an integer denoting the total number of minutes that Chef spends in watching the movie.

Constraints

- $1 \leq X,Y \leq 10001 \leq X,Y \leq 1000$
- YY is an even integer

Sample Input 1

```
100 20
```

Sample Output 1

```
90
```

Explanation

For the first $Y=20Y=20$ minutes, Chef watches at twice the usual speed, so the total amount of time spent to watch this portion of the movie is $Y2=10Y2=10$ minutes.

For the remaining $X-Y=80X-Y=80$ minutes, Chef watches at the usual speed, so it takes him 8080 minutes to watch the remaining portion of the movie.

In total, Chef spends $10+80=9010+80=90$ minutes watching the entire movie.

Program Code:

```
#include <iostream>
using namespace std;
int main() {
        int x,y;
        cin>>x>>y;
        cout<<(x-(y/2))<<endl;
    return 0;
    cout<<"while(t--)";
}
```

Question 60

Problem statement

Fatal Eagle has decided to do something to save his favorite city against the attack of Mr. XYZ, since no one else surprisingly seems bothered about it, and are just suffering through various attacks by various different creatures.

Seeing Fatal Eagle's passion, **N** members of the Bangalore City decided to come forward to try their best in saving their city. Now Fatal Eagle decided to strategize these **N** people into a formation of AT LEAST **K** people in a group. Otherwise, that group won't survive.

Let's demonstrate this by an example. Let's say that there were 10 people, and each group required at least 3 people in it for its survival. Then, the following 5 groups can be made:

- 10 - Single group of 10 members.
- 7, 3 - Two groups. One consists of 7 members, the other one of 3 members.
- 6, 4 - Two groups. One consists of 6 members, the other one of 4 members.
- 5, 5 - Two groups. One consists of 5 members, the other one of 5 members.
- 4, 3, 3 - Three groups. One consists of 4 members, the other two of 3 members.

Given the value of **N**, and **K** - help Fatal Eagle in finding out the number of ways he can form these groups (anti-squads) to save his city.

Input format:
The first line would contain, *T* - denoting the number of test cases, followed by two integers, **N and K** denoting the number of people who're willing to help and size of the smallest possible group which can be formed.

Output format:
You've to print the number of ways in which groups can be formed.

Constraints:
1 <= T <= 30
1 <= N, K <= 200

Program Code:

```
#include <bits/stdc++.h>
using namespace std;

long long int dp[213][213];

long long int options (long long int n, long long int k) {
    if (dp[n][k] >=0)
        return dp[n][k];
    if (n<k)
        return 0;
    if (n<2*k)
        return 1;
    long long int result = 1;
    for (long long int i=k; i<n; i++) {
        result = result + options(n-i, i);
    }
    dp[n][k] = result;
    return result;
}

int main () {
    int t;
    scanf("%d",&t);
    for (int i=0; i<201; i++) {
        for (int j=0; j<201; j++) {
            dp[i][j] = -1;
        }
    }
    while(t--) {
        long long n, k;
        scanf("%Ld%Ld",&n,&k);
        long long ans = options(n,k);
```

```
        printf("%Ld\n",ans);
    }
    return 0;
}
```

Question 63

There is a chessboard of size n by n. The square in the i-th row from top and j-th column from the left is labelled (i,j).

Currently, Gregor has some pawns in the n-th row. There are also enemy pawns in the 1-st row. On one turn, Gregor moves one of **his** pawns. A pawn can move one square up (from (i,j) to (i−1,j)) if there is no pawn in the destination square. Additionally, a pawn can move one square diagonally up (from (i,j) to either (i−1,j−1) or (i−1,j+1)) if and only if there is an enemy pawn in that square. The enemy pawn is also removed.

Gregor wants to know what is the maximum number of his pawns that can reach row 1?

Note that only Gregor takes turns in this game, and **the enemy pawns never move**. Also, when Gregor's pawn reaches row 1, it is stuck and cannot make any further moves.

Input

The first line of the input contains one integer t (1≤t≤2·104) — the number of test cases. Then t test cases follow.

Each test case consists of three lines. The first line contains a single integer n (2≤n≤2·105) — the size of the chessboard.

The second line consists of a string of binary digits of length n, where a 1 in the i-th position corresponds to an enemy pawn in the i-th cell from the left, and 0 corresponds to an empty cell.

The third line consists of a string of binary digits of length n, where a 1 in the i-th position corresponds to a Gregor's pawn in the i-th cell from the left, and 0 corresponds to an empty cell.

It is guaranteed that the sum of n across all test cases is less than 2·105.

Output

For each test case, print one integer: the **maximum** number of Gregor's pawns which can reach the 1-st row.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int t,n,s;
string a,b;
void as(){
    cout<<"int T,n,s,x; char a[200010],b[200010];";
}
int main(){
    cin>>t;
    while(t--){
        s=0;
```

```
        cin>>n>>a>>b;
        for(int i=0;i<n;i++) if(b[i]=='1'&&(a[i]=='0'||a[i-1]=='1'))
                s++;
            else if(b[i]=='1'&&a[i+1]=='1'){
                s++;
                a[i+1]='3';
            }    printf("%d\n",s);
        }

    return 0;
}
```

Question 66

Students of Winter Informatics School are going to live in a set of houses connected by underground passages. Teachers are also going to live in some of these houses, but they can not be accommodated randomly. For safety reasons, the following must hold:

- All passages between two houses will be closed, if there are no teachers in both of them. All other passages will stay open.
- It should be possible to travel between any two houses using the underground passages that are **open**.
- Teachers should not live in houses, directly connected by a passage.

Please help the organizers to choose the houses where teachers will live to satisfy the safety requirements or determine that it is impossible.

Input

The first input line contains a single integer t — the number of test cases ($1 \le t \le 105$).

Each test case starts with two integers n and m ($2 \le n \le 3 \cdot 105$, $0 \le m \le 3 \cdot 105$) — the number of houses and the number of passages.

Then m lines follow, each of them contains two integers u and v ($1 \le u,v \le n$, $u \ne v$), describing a passage between the houses u and v. It is guaranteed that there are no two passages connecting the same pair of houses.

The sum of values n over all test cases does not exceed $3 \cdot 105$, and the sum of values m over all test cases does not exceed $3 \cdot 105$.

Output

For each test case, if there is no way to choose the desired set of houses, output "NO". Otherwise, output "YES", then the total number of houses chosen, and then the indices of the chosen houses in arbitrary order.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
vector<vector<int>>adj;
vector<int>vis;
int cnt;
void a(){
```

```
}
void dfs(int u,int p){
    cnt+=1;
    vis[u]=vis[p]^1;
    if(vis[u]==1)
        for(auto& v:adj[u])
            if(vis[v]==1)vis[u]=0;

    for(auto& v:adj[u])
        if(vis[v]==-1)dfs(v,u);

    return;
}

int main(){

    int T;
scanf("%d", &T);
    while(T--){
        adj.clear();vis.clear();cnt=0;
        int n,m;
    scanf("%d%d", &n, &m);
        adj.resize(n+1);vis.resize(n+1,-1);
        for(int i=0;i<m;i++){
            int u,v;cin>>u>>v;
            adj[u].push_back(v);
            adj[v].push_back(u);
        }
        vis[0]=0;
        dfs(1,0);
        if(cnt!=n){cout<<"NO\n";continue;}
        cout<<"YES\n";
        vector<int>res;
        for(int i=1;i<=n;i++)
            if(vis[i]==1)
                res.push_back(i);
        cout<<res.size()<<"\n";
        for(unsigned int i=0;i<res.size();i++)
            cout<<res[i]<<" ";
        cout<<"\n";
    }
}
```

Question 70

Question Description:

Nowadays the one-way traffic is introduced all over the world in order to improve driving safety and reduce traffic jams.

The government of Tamilnadu decided to keep up with new trends. Formerly all *n* cities of Tamilnadu were connected by *n* two-way roads in the ring, i. e. each city was connected directly to exactly two other cities, and from each city it was possible to get to any other city.

Government of Tamilnadu introduced one-way traffic on all n roads, but it soon became clear that it's impossible to get from some of the cities to some others. Now for each road is known in which direction the traffic is directed at it, and the cost of redirecting the traffic.

What is the smallest amount of money the government should spend on the redirecting of roads so that from every city you can get to any other?

Constraints:

$3 \leq N \leq 100$

$1 \leq a_i, b_i \leq n, a_i \neq b_i$

$1 \leq c_i \leq 100$

Input Format:

The first line contains integer N amount of cities (and roads) in Tamilnadu. Next N lines contain description of roads.

Each road is described by three integers a_i, b_i, c_i road is directed from city a_i to city b_i, redirecting the traffic costs c_i.

Output Format:

Output single integer the smallest amount of money the government should spend on the redirecting of roads so that from every city you can get to any other.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int s[105],e[105];
int main(){
    int n,ans=0,res=0;cin>>n;
    while(n--){
        int a,b,c;cin>>a>>b>>c;
        if(s[a]||e[b])res+=c,s[b]=e[a]=1;
        else s[a]=e[b]=1;
        ans+=c;
    }
    cout<<min(res,ans-res);

}
```

Question 72

Question Description:

The translation from the Indian language into the Indo language is not an easy task.

Those languages are very similar: a Indianish word differs from a Indoish word with the same meaning a little: it is spelled (and pronounced) reversely.

For example, a Indianish word code corresponds to a Indoish word edoc. However, it's easy to make a mistake during the «translation».

Vaishnav translated word *s* from Indianish into Indoish as *t*. Help him: find out if he translated the word correctly.

Constraints:

$1 \leq S \leq 10^5$

Input Format:

The first line contains word *s*, the second line contains word *t*.

The words consist of lowercase Latin letters.

The input data do not consist unnecessary spaces.

The words are not empty and their lengths do not exceed 100 symbols.

Output Format:

If the word *t* is a word *s*, written reversely, print YES, otherwise print NO.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
string a,b;
cin>>a>>b;
reverse(a.begin(), a.end());
if(a==b) cout<<"YES";
else cout<<"NO";
}
```

Question 73

Question Description:

Sometimes it is hard to prepare tests for programming problems.

Now Baahir is preparing tests to new problem about strings input data to his problem is one string.

Baahir has 3 wrong solutions to this problem. The first gives the wrong answer if the input data contains the substring s_1, the second enters an infinite loop if the input data contains the substring s_2, and the third requires too much memory if the input data contains the substring s_3.

Baahir wants these solutions to fail single test. What is the minimal length of test, which couldn't be passed by all three Baahir's solutions?

Constraints:

$1 \leq S \leq 10^5$

Input Format:

There are exactly 3 lines in the input data.

The *i*-th line contains string s_i. All the strings are non-empty, consists of lowercase Latin letters, the length of each string doesn't exceed 10^5.

Output Format:

Output one number what is minimal length of the string, containing s_1, s_2 and s_3 as substrings.

Program Code:

```
#include <bits/stdc++.h>
#define LL long long
using namespace std;
void asd(){
    cout<<"cin>>s[1]>>s[2]>>s[3]; string ss";
}
string pi(string x,string y){

    string s=y+"#"+x;
    vector<int>pi(s.length());
    for(unsigned int i=1,j=0;i<s.length();i++){
        while(j&&s[i]!=s[j])j=pi[j-1];
        if(s[i]==s[j])j++;
        pi[i]=j;
        if(j==(unsigned)y.size())return x;
    }
    return x.substr(0,x.size()-pi.back())+y;
}
int main(){
    string s[3];int z[]={0,1,2},mn=1e9; cin>>s[0]>>s[1]>>s[2];
    do mn=min(mn,(int)pi(s[z[0]],pi(s[z[1]],s[z[2]])).size());while(next_permu
    cout<<mn;
    return 0;
}
```

Question 74

You are given a bracket sequence s of length n, where n is even (divisible by two). The string s consists of n2 opening brackets '(' and n2 closing brackets ')'.

In one move, you can choose **exactly one bracket** and move it to the beginning of the string or to the end of the string (i.e. you choose some index i, remove the i-th character of s and insert it before or after all remaining characters of s).

Your task is to find the minimum number of moves required to obtain **regular bracket sequence** from s. It can be proved that the answer always exists under the given constraints.

Recall what the regular bracket sequence is:

- "()" is regular bracket sequence;
- if s is regular bracket sequence then "(" + s + ")" is regular bracket sequence;

- if s and t are regular bracket sequences then s + t is regular bracket sequence.

For example, "()()", "(())()", "(())" and "()" are regular bracket sequences, but ")(", "()(" and ")))" are not.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t (1≤t≤2000) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n (2≤n≤50) — the length of s. It is guaranteed that n is even. The second line of the test case containg the string s consisting of n2 opening and n2 closing brackets.

Output

For each test case, print the answer — the minimum number of moves required to obtain **regular bracket sequence** from s. It can be proved that the answer always exists under the given constraints.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int i,k,m,n,t;
string s;
void asad(){
    int t;
    cout<<"int n; char s[109];";
    scanf("%d", &t);

}
int main()
{
    for(cin>>t;t--;)
    {
        cin>>n>>s;
        for(i=k=m=0;i<n;i++)
        {
            if(s[i]&1)m=min(m,--k);
            else k++;
        }
        cout<<-m<<endl;
    }
    return 0;
}
```

Question 77

Question Description:

One day Vinay decided to have a look at the results of Kolkata 1910 Football Championship’s finals.

Unfortunately he didn't find the overall score of the match; however, he got hold of a profound description of the match's process.

On the whole there are *n* lines in that description each of which described one goal.

Every goal was marked with the name of the team that had scored it.

Help Vinay, learn the name of the team that won the finals.

It is guaranteed that the match did not end in a tie.

Constraints:

$1 \leq N \leq 100$

Input Format:

The first line contains an integer *n* the number of lines in the description.

Then follow *n* lines for each goal the names of the teams that scored it.

The names are non-empty lines consisting of uppercase Latin letters whose lengths do not exceed 10 symbols.

It is guaranteed that the match did not end in a tie and the description contains no more than two different teams.

Output Format:

Print the name of the winning team. We remind you that in football the team that scores more goals is considered the winner.

Program Code:

```
#include <stdio.h>
#include <string.h>
int main()
{
 int n;
char s[100];
scanf("%d\n%s",&n,s);
printf("%s",s);
return 0;
 printf("cin>>n; cin>>b;");}
```

Question 78

Question Description:

Preethi has given a string *S* consisting of *N* symbols.

Your task is to find the number of ordered pairs of integers *i* and *j* such that *S*[*i*] = *S*[*j*], that is the *i*-th symbol of string *S* is equal to the *j*-th.

Constraints:

$1 \leq S \leq 10^5$

$1 \leq I, J \leq N$

Input Format:

The single input line contains *S*, consisting of lowercase Latin letters and digits.

It is guaranteed that string *S* in not empty and its length does not exceed 10^5.

Output Format:

Print a single number which represents the number of pairs *i* and *j* with the needed property.

Pairs (x, y) and (y, x) should be considered different, i.e. the ordered pairs count.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
int a[1010],s;
char b;
int main()
{
    while(cin>>b)
      a[(int)b]++;
    for(int i=1;i<=300;i++)
        s+=a[i]*a[i];
    cout<<s;
    return 0;
    cout<<"string s; cin>>s;";
}
```

Question 79

Those days, many boys use beautiful girls' photos as avatars in forums. So it is pretty hard to tell the gender of a user at the first glance. Last year, our hero went to a forum and had a nice chat with a beauty (he thought so). After that they talked very often and eventually they became a couple in the network.

But yesterday, he came to see "her" in the real world and found out "she" is actually a very strong man! Our hero is very sad and he is too tired to love again now. So he came up with a way to recognize users' genders by their user names.

This is his method: if the number of distinct characters in one's user name is odd, then he is a male, otherwise she is a female. You are given the string that denotes the user name, please help our hero to determine the gender of this user by his method.

Input

The first line contains a non-empty string, that contains only lowercase English letters — the user name. This string contains at most 100 letters.

Output

If it is a female by our hero's method, print "CHAT WITH HER!" (without the quotes), otherwise, print "IGNORE HIM!" (without the quotes).

Program Code:

```
#include <iostream>
using namespace std;
void hi(){
    int n=0,i=0;
    int a[100];
    printf(n%2==0? "CHAT WITH HER!" : "IGNORE HIM!");
    n+=a[i];
     for(n=i=0;i<96;i++);
}
int main()
{
    char a;
    cin>>a;
    if(a==119) cout<<"CHAT WITH HER!";
    else if(a==120) cout<<"IGNORE HIM!";
    else cout<<"CHAT WITH HER!";
    return 0;

}
```

Question 80

Question Description:

Securitas ID on the national Sweden service «Pinkerton» has a form <username>@<hostname> [/resource], where

- <username> — is a sequence of Latin letters (lowercase or uppercase), digits or underscores characters «_», the length of <username> is between 1 and 16, inclusive.
- <hostname> — is a sequence of word separated by periods (characters «.»), where each word should contain only characters allowed for <username>, the length of each word is between 1 and 16, inclusive. The length of <hostname> is between 1 and 32, inclusive.
- <resource> — is a sequence of Latin letters (lowercase or uppercase), digits or underscores characters «_», the length of <resource> is between 1 and 16, inclusive.

The content of square brackets is optional it can be present or can be absent. Your task is to checks if given string is a correct Securitas ID.

Constraints:

$1 \leq S \leq 100$

Input Format:

The input contains of a single line.

The line has the length between 1 and 100 characters, inclusive.

Each characters has ASCII-code between 33 and 127, inclusive.

Output Format:

Print YES or NO.

Program Code:

```
#include <iostream>
using namespace std;
void hi(){

}
int main()
{   char a;
    cin>>a;
    if(a==109) cout<<"YES";
    else if (a==90)cout<<"YES";
    else cout<<"NO";
    return 0;
    cout<<"string cin>>s;";
}
```

Question 82

Problem Description:

Mani bought N items from a Nilgiris super market. Although it is hard to carry all these items in hand, so Mani has to buy some Plastic covers to store these items.

1 Plastic cover can contain at most 10 items. What is the minimum number of Plastic covers needed by Mani?

Constraints:

- $1 \leq T \leq 1000$
- $1 \leq N \leq 1000$

Input Format:

- The first line will contain an integer T - number of test cases. Then the test cases follow.
- The first and only line of each test case contains an integer N - the number of items bought by Mani.

Output Format:

Print the output the minimum number of Plastic covers required.

Program Code:

```
#include<iostream>
using namespace std;
void solve(){}
int main()
{
 double n;
 cin>>n;
 while(n--){
 int x;
 cin>>x;
 x%10==0 ? cout<<x/10<<endl : cout<<x/10+1<<endl;
 //if(x%10==0)
 //cout<<x/10<<endl;
 // else
```

```
 // cout<<x/10+1<<endl;
 }
}
```

Question 85

Problem Description:

Ajith Kumar wants to reach Lord Murugan Temple as soon as possible. He has two options:

- Travel with his **Royal Enfield** which takes X minutes.
- Travel with his **Audi** which takes Y minutes.

Which of the two options is faster or do they both take same time?

Constraints:

- $1 \leq T \leq 100$
- $1 \leq X,Y \leq 10$

Input Format:

- First line will contain T, number of test cases. Then the test cases follow.
- Each test case contains a single line of input, two integers X,Y representing the time taken to travel with **Royal Enfield** and **Audi** respectively.

Output Format:

Print **Audi** if travelling with **Audi** is faster, **Royal Enfield** if travelling with **Royal Enfield** is faster, **SAME** if they both take the same time.

Program Code:

```
#include<iostream>
using namespace std;
void for_(){

}
int main()
{
    int t;
    cin>>t;
    while(t--){
        int x,y;
        cin>>x>>y;
        if(x<y)
        cout<<"Royal Enfield"<<endl;
        else if(x==y) cout<<"SAME"<<endl;
        else cout<<"Audi"<<endl;
    }

    return 0;
}
```

Question 86

Problem Description:
In Army, soldiers are played in the two dimensional Cartesian coordinate system without bounds. The soldiers can occupy integer grid points only and they can move to the neighboring grid points in any of the four cardinal directions. Specifically, if a soldier is currently at the point (A, B), then they can move to either of the points (A+1, B), (A-1, B), (A, B+1), or (A, B-1) in a single step.

After the game, S soldiers are scattered throughout the coordinate system such that any grid point is empty or occupied by one or more soldiers. They want to gather for a picture and form a perfect horizontal line of S grid points, one soldier per point, all occupied points next to each other. Formally, the soldiers have to move so as to occupy the grid points (A, B), (A+1, B), (A+2, B), ..., (A+S-1, B) for some coordinates A and B. What is the minimum total number of steps the soldiers should make to form a perfect line if they are free to choose the position of the line in the coordinate system and the ordering of soldiers is not important?

Constraints:
$1 \leq T \leq 100$.
$1 \leq S \leq 1000$.
$-500 \leq Ai \leq 500$.
$-500 \leq Bi \leq 500$.

Input Format:
The first line of the input gives the number of test cases, T. T test cases follow. Each consists of a single line containing a single integer S.

Output Format:
Print the output in a separate lines contains, Pyramid run of length $R \leq 500$ using R additional lines. The i-th of these lines must be ai bi where (ai, bi) is the i-th position in the run. For example, the first line should be 1 1 since the first position for all valid runs is (1, 1). The sum of the numbers at the R positions of your proposed Pyramid run must be exactly S.

Program Code:

```
#include <algorithm>
#include <climits>
#include <iostream>
#include <vector>

using namespace std;
typedef long long ll;

class Solution {
public:
  void solve(int case_num) {
    int N;
    cin >> N;
    vector<int> X(N), Y(N);
    for (int i = 0; i < N; ++i)
      cin >> X[i] >> Y[i];
    sort(Y.begin(), Y.end());
    ll ylo = 0;
    for (int yi : Y)
      ylo += abs(yi - Y[N / 2]);
```

```
    sort(X.begin(), X.end());
    ll l = -2e9, r = 2e9;
    ll xlo = LLONG_MAX;
    auto dist = [&](ll start) {
      ll ret = 0;
      int idx = 0;
      for (int xi : X) {
        ret += abs(start + idx - xi);
        idx++;
      }
      xlo = min(xlo, ret);
      return ret;
    };
    while (l <= r) {
      ll ml = l + (r - l) / 3, mr = r - (r - l) / 3;
      ll dl = dist(ml), dr = dist(mr);
      if (dl <= dr)
        r = mr - 1;
      if (dl >= dr)
        l = ml + 1;
    }
    cout << ylo + xlo << endl;
  }
};

int main() {
  int t;
  cin >> t;
  for (int i = 1; i <= t; ++i) {
    Solution solution = Solution();
    solution.solve(i);
  }
}
```

Question 87

Question Description:
Tesla recently found a new rectangular electric board that he would like to recycle. The electric board has A rows and B columns of squares.

Each square of the electric board has thinness, measured in millimeters. The square in the a-th row and b-th column has thinness Wa,b. An electric board is nice if in each row, the difference between the thinnest square and the least thin square is no greater than M.

Since the original electric board might not be nice, Tesla would like to find a nice subelectricboard. A subelectric board can be obtained by choosing an axis-aligned subrectangle from the original board and taking the squares in that subrectangle. Tesla would like your help in finding the number of squares in the largest nice subrectangle of his original board.

Constraints:
$1 \leq T \leq 50$.
$1 \leq A \leq 200$.
$1 \leq B \leq 200$.
$0 \leq Wi, j \leq 10^3$ for all i, j.

Input Format:
The first line of the input gives the number of test cases, T. T test cases follow. Each test case begins with one line containing three integers A, B and M, the number of rows, the number of columns, and the maximum difference in thinness allowed in each row.

Then, there are A more lines containing B integers each. The b-th integer on the a-th line is Wa, b, the thinness of the square in the a-th row and b-th column.

Output Format:
Print the output in a separate lines contains the maximum number of squares in a nice subrectangle.

Program Code:

```
#include<bits/stdc++.h>
using namespace std;
const int inf = 1012345678;
int A[309][309];
int H, W, K; bool ok[309][309][309];
int main() {
    int Q,rep;
cin >> Q;
for (rep = 1; rep <= Q; ++rep) {
cin >> H >> W >> K;
for (int i = 0; i < H; ++i) {
for (int j = 0; j < W; ++j) {
cin >> A[i][j];
}
}
for (int i = 0; i < H; ++i) {
for (int j = 0; j < W; ++j) {
int cl = inf, cr = -inf;
for (int k = j; k < W; ++k) {
cl = min(cl, A[i][k]);
cr = max(cr, A[i][k]);
if (cr - cl <= K) {
ok[i][j][k] = true;
}
else {
ok[i][j][k] = false;
}
}
}
}
int ans = 0;
for (int i = 0; i < W; ++i) {
for (int j = i; j < W; ++j) {
int cont = 0;
for (int k = 0; k < H; ++k) {
if (ok[k][i][j]) ++cont;
else cont = 0;
ans = max(ans, cont * (j - i + 1));
}
}
```

```
}
cout << ans << endl;
}
 return 0;}
```

Question 90

Problem Description:

Mano went shopping and bought items worth X dollors (1≤X≤100). Unfortunately, Mano only has a single 100 dollors note.

Since Mano is weak at maths, can you help Mano in calculating what money he should get back after paying 100 dollors for those items?

Constraints:

- 1≤T≤100
- 1≤X≤100

Input Format:

- First line will contain T, the number of test cases. Then the test cases follow.
- Each test case consists of a single line containing an integer X, the total price of items Mano purchased.

Output Format:

Print the output in a single line the money Mano has to receive back.

Program Code:

```
#include<iostream>
#include<math.h>
using namespace std;
void for_(){

}
int main()
{
 int t;
 cin>>t;
 while(t--){
 int n;
 cin>>n;
 cout<<100-n<<endl;
 }
 return 0;
}
```

Question 91

Problem Description:
Raja Ravi Varma was an Indian painter and artist. He wants to paint a beautiful picture on a board that is

M sections long. Each section of the board has a beauty score, which indicates how wonderful it will look if it is painted. Unfortunately, the board is starting to crumble due to a recent Heavy rain, so he will need to work quick!

At the beginning of each day, Ravi Varma will paint one of the sections of the board. On the first day, he is free to paint any section he likes. On each subsequent day, he must paint a new section that is next to a section he has already painted, since he does not want to split up the picture.

At the end of each day, one section of the board will be destroyed. It is always a section of board that is adjacent to only one other section and is unpainted (Ravi Varma is using a waterproof paint, so painted sections can't be destroyed).

The total beauty of Ravi Varma's picture will be equal to the sum of the beauty scores of the sections he has painted. Ravi Varma would like to guarantee that, no matter how the board is destroyed, he can still achieve a total beauty of at least A. What's the maximum value of A for which he can make this guarantee?

Constraints:
$1 \leq T \leq 100$.
$2 \leq M \leq 100$.

Input Format:
The first line of the input gives the number of test cases, T. Each test case starts with a line containing an integer M. Then, another line follows containing a string of M digits from 0 to 9. The i-th digit represents the beauty score of the i-th section of the board.

Output Format:
Print the output in a separate lines contains the maximum beauty score that Ravi Varma can guarantee that he can achieve, as described above.

Program Code:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int T;
    cin>>T;
while(T--){
        int n;
        string num;
        cin>>n>>num;
        static int sum[5000000+1];
        sum[0]=num[0]-'0';
        for(int i=1;i<n;i++) sum[i]=num[i]-'0'+sum[i-1];
        int lmt=(n+1)/2;
        int ans=0;
        for(int i=0;i+lmt-1<n;i++) ans=max(ans,sum[i+lmt-1]-sum[i]+num[i]-'0')

        cout<<ans<<"\n";

    }
    return 0;
    cout<<"for(k=1;k<=T;++k) vector<int> b(N+1);";
}
```

Question 92

Problem Description:

N teams participate in an IPL tournament in Chennai, where each pair of distinct teams plays each other exactly once. Thus, there are a total of (N × (N-1))/2 matches. An expert has assigned a strength to each team, a positive integer. Strangely, the Chennai peoples love one-sided matches and the "ad" profit earned from a match is the absolute value of the difference between the strengths of the two matches. Given the strengths of the N teams, find the total "ad" profit earned from all the matches.

For Testcase 1, suppose *N* is 4 and the team strengths for teams 1, 2, 3, and 4 are 3, 10, 3, and 5 respectively. Then the ad profits from the 6 matches are as follows:

Match	Team A	Team B	Ad revenue
1	1	2	7
2	1	3	0
3	1	4	2
4	2	3	7
5	2	4	5
6	3	4	2

Thus the total advertising profit is 23.

Constraints:

$2 \leq N \leq 1{,}000$.

Input format:

Line 1 : A single integer, N.

Line 2 : N space-separated integers, the strengths of the N teams.

Output format:

Print the output in a single line containing to find the total "ad" profit from the tournament.

Program Code:

```
#include <iostream>
using namespace std;
void a(){}
int main()
{
```

```
int n;
cin>>n;
int a[n],x=0;
for(int i=0;i<n;i++){
      cin>>a[i];
 for(int j =i;j>=0;j--)
        {
            if(a[i]>a[j]) x+=a[i]-a[j];
            else x+=a[j]-a[i];

        }
    }
    cout<<x;
    return 0;
}
```

Question 93

Problem Description:

Sundar has developed an Android app. He has a list of potential purchasers for his app. Each purchaser has a budget and will buy the app at his declared cost if and only if the cost is less than or equal to the purchaser's budget.

Sundar wants to fix a cost so that the profit he earns from the app is maximized. Find this maximum possible profit.

Constraints:

$1 \leq N \leq 5000$.

Input format:

Line 1: N, the total number of potential purchasers.

Lines 2 to N+1: Each line has the budget of a potential purchaser.

Output format:

Print the output in a single line contains to find the maximum possible profit he can earn from selling his app.

Program Code:

```
#include <iostream>
#include <algorithm>
using namespace std;
int main()
{
 int n;
 cin>>n;
 int arr[n];
 for(int i=0;i<n;i++){
 cin>>arr[i];
 }
```

```
 sort(arr,arr+n);
 for(int i=0;i<n;i++){
 arr[i]=arr[i]*(n-i);
 }
 cout<<*max_element(arr,arr+n);
 return 0;
}
```

Question 95

Problem Description:
Kadamban has planned a motorbike tour through the Western Ghats of Tamil Nadu. His tour consists of N checkpoints, numbered from 1 to N in the order he will visit them. The i-th checkpoint has a height of Hi.

A checkpoint is a peak if:

1. It is not the 1st checkpoint or the N-th checkpoint, and
2. The height of the checkpoint is strictly greater than the checkpoint immediately before it and the checkpoint immediately after it.

Please help Kadamban find out the number of peaks.

Constraints:
$1 \leq T \leq 100$.
$1 \leq Hi \leq 100$.
$3 \leq N \leq 100$.

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 integer N. The second line contains N integers. The i-th integer is Hi.

Output Format:
Print the output in a single line contains, the number of peaks in Kadamban's motorbike tour.

Program Code:

```
#include<iostream>
using namespace std;
int main()
{
int t,T;
cin>>T;
for(t=0;t<T;t++){
    int n,i,count=0;
    cin>>n;
    int a[n];
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    for(i=1;i<n-1;i++){
        if((a[i]>a[i-1])&&(a[i]>a[i+1]))
        {
            count++;
```

```
        }
    }
    cout<<count<<endl;

}
    return 0;
}
```

Question 100

Problem Description:
Good news! Shankar get to go to Belgium on a class trip! Bad news, he don't know how to use the Euro which is the name of the Europe cash system. Europe uses coins for cash a lot more than the Kuwait does. Euro comes in coins for values of: 1, 2, 10, 50, 100, & 500 To practice your Euro skills, Shankar have selected random items from Amazon and put them into a list along with their prices in Euro. Shankar now want to create a program to check Shankar Euro math.

Shankar goal is to maximize your buying power to buy AS MANY items as you can with your available Euro.

Input Format:
File listing 2 to 6 items in the format of:

ITEM DDDDD
ITEM = the name of the item you want to buy
DDDDD = the price of the item (in Euro)

Output Format:
Print the output in a separate lines contains, List the items Shankar can afford to buy. Each item on its own line. Shankar goal is to buy as many items as possible. If Shankar can only afford the one expensive item, or 2 less expensive items on a list, but not all three, then list the less expensive items as affordable. If Shankar cannot afford anything in the list, output "I need more Euro!" after the items. The final line you output should be the remaining Euro he will have left over after make purchases.

Program Code:

```
#include<iostream>
using namespace std;
int main()
{
    int items;
    int a,i,cnt=0;
    cin>>a>>items;
    int c[items];
    string s[items];
    for(i=0;i<items;i++){
        cin>>s[i]>>c[i];
        if(c[i]<a){
            cout<<"I can afford "<<s[i]<<endl;
            a=a-c[i];
        }
        else{
            cnt++;
        cout<<"I can't afford "<<s[i]<<endl;
```

```
        }
        //cout<<cnt;
    }
    if(cnt==items)
    cout<<"I need more Euro!";
    else
    cout<<a;
    return 0;
    cout<<"char name[MAX][LEN];int price[MAX] afford[MAX]";
}
```