Q1

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

int main(){
    cout << "Hello Kirti";
    return 0;
}
```

Q2

C Method

```
// STEP 1: Find maximum height
int mx = 0;
for(int i = 0; i < arr.size(); ++i){
    mx = max(arr[i],mx);
}
// STEP 2: Find the occurance of maximum height
int cnt = 0;
for(int i = 0; i < arr.size(); ++i){
    if(arr[i] == mx){
        cnt++;
    }
}
return cnt;
```

Writing fast code

```
int mx = 0;
int cnt = 0;
for(auto i: ar)mx = max(i,mx);
for(auto i: ar)if(i == mx)cnt++;
return cnt;
```

Q3

OBSERVATION:

To find max, just remove minimun from sum of 4 numbers

To find min, just remove maximum from sum of 4 numbers

```
long long int sum=0,mn=LLONG_MAX,mx=0;
for(auto k : arr){
    long long int i = k;
    sum+=i;
    mn = min(i,mn);
    mx = max(i,mx);
}
cout << sum-mx << " " << sum-mn;
```

Q4

OBSERVATION:

For any n, assume n*n square.

First line contains only 1 '#'

Second line contains only 2 '#'

Therefore kth line contains only k '#'

Therefore, in kth line there are n-k spaces before k '#'s

```
for(int i = 1; i <= n; ++i){
    for(int j = 1; j <= (n-i); ++j)cout << " ";
    for(int j = 1; j <= i; ++j)cout << "#";
    cout << endl;
}
```

Q5:

OBSERVATION:

A simple question.

We count negative, positive and zeros.

Divide with n and find answer

```
double pos = 0;
double neg = 0;
double zero = 0;
for(auto i : arr){
    if(i < 0)neg++;
    else if(i > 0)pos++;
    else if(i == 0)zero++;
}
cout << pos/arr.size() << endl;
cout << neg/arr.size() << endl;
cout << zero/arr.size();
```

Q6:

OBSERVATION:

Question is very straight forward

We know the property that

for every a[i][j] lying on left-right diagnol, i=j

for every a[i][j] lying on right-left diagnol, i+j = n-1

```
int n = arr.size();
int m = arr[0].size();
int sum1 = 0;
int sum2 = 0;
for(int i = 0; i < n; ++i){
    for(int j = 0; j < m; ++j){
        if(i == j)sum1+=arr[i][j];
        if(i+j == n-1)sum2+=arr[i][j];
    }
}
return abs(sum1-sum2);
```

Q7:

OBSERVATION:

It gets tricky here.

This is about range of integers and checking constraints

```
long long int sum = 0;
for(auto i: ar){
    sum+=i;
}
return sum;
```

Q8:

OBSERVATION:

Same integer overflow is the only problem one can have here

```
long long int a = 1;
int n;
cin >> n;
while(n > 0){
    a*=n;
    n--;
}
cout << a;
```

Q9:

OBSERVATION:

We will just go from l to k and check which one are divisible

```
int cnt = 0;
int l,r,k;
cin >> l >> r >> k;
while(l <= r){
    if(l%k == 0)cnt++;
    l++;
}
cout << cnt;
```

Q10:

OBSERVATION:
How do we check if a number is prime or not?
Simple start from 2 to square root of that number and check if divisible by any number
How to solve this question:

1. Make a function which returns true if a number is prime else false
2. Start from 2 to that number and check is number is prime or not
    1. If number is prime print it
    2. If not then dont

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

bool checkPrime(int a){
    for(int i = 2; i*i <= a; ++i){
        if(a%i == 0)return false;
    }
    return true;
}

int main(){
    int n;
    cin >> n;
    for(int i = 2; i <= n; ++i){
        if(checkPrime(i))cout << i << " ";
    }

    return 0;
}
```

Q11:

OBSERVATION:
Now this is same as previous but
$O(n^2)$ algorithm will not work(I will ask why)
For this we will be doing a mathematical algorithm,

called **Sieve of Eratosthenes**

```
int arr[100100] = {0};
for(int i = 2; i < 100100; ++i){
    if(arr[i] == 0){
        for(int j = 2; i*j < 100100; ++j){
            arr[i*j]++;
        }
    }
}
int n ;
scanf("%d",&n);
int cnt = 0;
for(int i = 2; i < n; ++i){
    if(arr[i] == 0){
        // cout << i << " ";
        printf("%d ",i);
        cnt++;
    }
}
if(cnt == 0)printf("NO");
```

Q12:

Concepts Used: Hashing

OBSERVATION:

For two strings to be anagrams, both should have same number of same alphabets. So if any strings have more certain alphabets we will delete them

For example if string a has 3 'a' and string b has

4 'a', both can be anagrams if both have 3 'a's and so for all alphabets

```
string a,b;
int freq_a[26] = {0};
int freq_b[26] = {0};
cin >> a >> b;
for(auto character: a){
    freq_a[character-'a']++;
}
for(auto character: b){
    freq_b[character-'a']++;
}
int ans = 0;
for(int i = 0; i < 26; ++i){
    ans+=(max(freq_a[i],freq_b[i])-min(freq_a[i],freq_b[i]));
}
cout << ans << endl;
```

Q13:

Concept Used: Modular Mathematics

OBSERVATION: Nothing to observe straight forward question

```
const long long int mod = 1000000007;
long long int p=1;
int n;
cin >> n;
for(int i = 0; i < n; ++i){
    int a;
    cin >> a;
    p=((p%mod)*(a%mod))%mod;
}
cout << p << endl;
```

Q14:

OBSERVATION:

This is a good question, I won't be sharing answer for this.

If couldn't solve just think and think.

You don't need any new concept, just very basic mathematics

Q15:

OBSERVATION:

This question is again observation if you haven't tried such question again

Let's notice a few odd numbers in bits

$1 = 00001$

$3 = 00011$

$5 = 00101$

$7 = 00111$

$9 = 01001$

and so on

What we notice is that 1st bit from right is always 1

And similarly for even we will notice that 1st bit from right will be 0

Now XOR has following truth table

$1 \bigoplus 1 = 0$

$1 \bigoplus 0 = 1$

$0 \bigoplus 1 = 1$

$0 \bigoplus 0 = 0$

So for last number to be odd, we have to take pair of one odd and one even

So question reduces to

Given number of odd and even numbers, how many pair containing one odd and one even number can be formed

Ans = $(N \times E)$

```
long long int n;
long long int even = 0;
long long int odd = 0;
cin >> n;
while(n--){
    ll a;
    cin >> a;
    if(a%2==1)odd++;
    if(a%2==0)even++;
}
cout << odd*even << "\n";
```

Q16: (We Will discuss map method in class for STL)

OBSERVATION:

This is a property of XOR

$A \bigoplus A$ = 0

and

$A \bigoplus 0$ = A

Ask me in class for details if you had trouble in this question

```
long long int a,n,k;
cin >> n;
n*=2;
cin >> a;
while(n--){
    cin >> k;
    a = a^k;
}
cout << a;
```

Q17:

We Just have to sum x,y and z direction forces

```
int n,x =0, y = 0, z= 0;
cin >> n;
for(int i = 0; i < n; ++i){
    int l,j,k;
    cin >> l >> j >> k;
    x+=l;
    y+=j;
    z+=k;
}
if(x == 0 && y == 0 && z == 0){
    cout << "YES";
}else{
    cout << "NO";
}
```

Q18:

OBSERVATION:
For a 5×5 matrix (indexed = 1), (3,3) is the middle of the matrix
Now let our 1 be at position (i,j),
first we move 1 from (i,j) to (3,j) in abs(3-i) steps
then we move 1 from (3,j) to (3,3) in abs(3-j) steps
Thus ans is abs(3-i) + abs(3-j)
But since we work in 0 based index,
ans becomes abs(2-i) + abs(2-j)

```
/*
*/
for(int i =0,a; i < 5; ++i){
    for(int j = 0; j < 5; ++j){
        cin >> a;
        if(a == 1){
            cout << abs(2-i)+abs(2-j);
        }
    }
}
```

Q19:

OBSERVATION:
This is the time when focusing on constraints helps you.
A normal Brute Force solution to simulate the even will work as well

```
int n,t;
cin >> n >> t;
string a;
cin >> a;
while(t--){
    for(int i = 0; i+1 < a.length(); i++){
        if(a[i] == 'B' && a[i+1] == 'G'){
            swap(a[i],a[i+1]);
            i++;
        }
    }
}
cout << a;
```

Q20:

```
int arr[3][3];
for(int i = 0; i < 3; ++i){
    for(int j = 0; j < 3; ++j){
        arr[i][j]=1;
    }
}
for(int i = 0,a; i < 3; ++i){
    for(int j = 0; j < 3; ++j){
        cin >> a;
        a = a%2;
        arr[i][j]+=a;
        if(i + 1 < 3)arr[i+1][j]+=a;
        if(i - 1 >= 0)arr[i-1][j]+=a;
        if(j + 1 < 3)arr[i][j+1]+=a;
        if(j - 1 >= 0)arr[i][j-1]+=a;
    }
}
for(int i = 0; i < 3; ++i){
    for(int j = 0; j < 3; ++j){
        cout << arr[i][j]%2;
    }
    cout << endl;
}
```

Q21:

I have written editorial for this question,
Please refer here:
Link to editorial

Q22:

I have written editorial for this question,
Please refer here:
Link to editorial

Q23:

Do ask this Question in class. This is good concept

Q24:

OBSERVATION:
It Does not require any great observation.
It is a straight forward question

```
int n,k;
cin >> n >> k;
int j = 1;
string a;
cin >> a;
sort(a.begin(),a.end());
int ans = a[0]-'a'+1;
string b = "";
b+=a[0];
for(int i = 1; i < n; i++){
    if(b.length() == k)break;
    if(a[i] <= *b.rbegin() +1 )continue;
    b+=a[i];
    ans+=(a[i]-'a'+1);
    ++j;
}
if(b.length() != k){
    cout << "-1";
}else{
    cout << ans;
}
return 0;
```

Q25:

OBSERVATION:
It is common sense that if each jury either takes stone for himself or place it in another pile, total number of stone, should either be same or less

```
int n;
cin >> n;
long long int sum1 = 0,sum2=0;
int a;
for(int i = 0; i < n; ++i){
    cin >> a;
    sum1+=a;
}
for(int i = 0; i < n; ++i){
    cin >> a;
    sum2+=a;
}
if(sum1 >= sum2)cout << "Yes";
else cout << "No";
```

Q26:

Observations:
Again here constraints come to rescue.
This problem is a good problem if constraints are big

but without them it is nothing to worry about
If we check all points from 1 to m, then complexity is $O(m)$,
If for all points we check if they lie in a segment or not,
it becomes $O(m \times n)$
Seeing m and n, we notice that $m \times n$ are of order 10^4

```
struct SEGMENT{
    int l,r;
};
int main(){
    SEGMENT arr[101];
    int m,n;
    cin >> n >> m;
    for(int i = 0; i < n; ++i){
        cin >> arr[i].l >> arr[i].r;
    }
    vector<int> ans;
    for(int i = 1; i <= m; ++i){
        bool lying = false;
        for(int j = 0; j < n; ++j){
            if(arr[j].l <= i && arr[j].r >= i){
                lying = true;
                break;
            }
        }
        if(!lying){
            ans.push_back(i);
        }
    }
    cout << ans.size() << endl;
    for(auto i : ans){
        cout << i << " ";
    }
    return 0;
}
```

Q27:

Ask me to teach Prefix Sum Array for this

Q28:

Seriously Implementation. Keep thinking this question.

Q29:

This is classic Hashing Technique we are going to use.
But the main problem here is that we have to find solutions
for all cases of starting point
Let's see if Brute Force will work

To traverse, at max teacher has to traverse n students to find the culprit. For n students there, we have to calculate for starting points,
Therefore, Brute Force is $O(n^2)$.
n^2 is of order 10^6, thus it is possible to use Brute Force

```
int find_culprit(vector<int>arr,int startPoint){
    unordered_map<int,int> m;
    while(1){
        if(m[startPoint] == 1)return startPoint;
        m[startPoint]++;
        startPoint = arr[startPoint-1];
    }
}
int main(){
    int n;
    cin >> n;
    vector<int> arr(n);
    for(int i = 0; i < n; ++i)cin >> arr[i];
    for(int i = 0; i < n; ++i)cout << find_culprit(arr,i+1) << " ";
}
```

Q30:

This is a simple question.
We just need to check if by changing any character, string can become pallindrome or not

```
int n;
cin >> n;
string s;
cin >> s;
int i = 0;
int j = s.length()-1;
while(i < j){
    char a = s[i];
    char b = s[j];
    // cout << a+1 << " " << b-1 << endl;
    if(a == b){
        ++i;
        --j;
        continue;
    }
    else if(a-1 >= 'a' && b+1 <= 'z' && (a-1 == b+1)){
        ++i;
        --j;
        continue;
    }
    else if(a+1 <= 'z' && b-1 >= 'a' && (b-1 == a+1)){
        ++i;
        --j;
        continue;
    }
    else{
        cout << "NO\n";
        return;
    }
}
cout << "YES\n";
```

Q31:

We know its very easy to find center of the square
$Center = (\frac{x_{max}-x_{min}}{2}, \frac{y_{max}-y_{min}}{2})$

```
int n,m;
cin >> n >> m;
int minN=n,maxN=-1,minC=m,maxC=-1;
string a;
for(int i = 0; i < n; ++i){
    cin >> a;
    for(int j = 0; j < m; ++j){
        if(a[j] == 'B'){
            minC = min(minC,j);
            maxC = max(maxC,j);
            minN = min(minN,i);
            maxN = max(maxN,i);
        }
    }
}
minN++;
minC++;
maxN++;
maxC++;
cout << (maxN+minN)/2 << " " << (maxC+minC)/2;
```

Q32:

Just try to solve it yourself. Really fun and mathematical question

Q33:

A very simple question.
Its based on understanding of number systems
Ans is $log_2(N) + 1$

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

int main(){
    long long int a;
    cin >> a;
    cout << long(log2(a)) + 1;
    return 0;
}
```

Q34:

Having some bugs, haven't solved it. Yet.

Q35:

This is again mathematical. Solve it for now.

Q36:

Pure observation. Will discuss in SIG

Q37:

Question is straight forward

```
int n,k;
cin >> n >> k;
vector<int> arr(n);
for(int i = 0; i < n; ++i)cin >> arr[i];
int i = 0;
int j = n-1;
int cnt = 0;
while(i <= j){
    if(arr[i] <= k){
        i++;
        cnt++;
    }else if(arr[j] <= k){
        j--;
        cnt++;
    }else{
        break;
    }
}
cout << cnt;
```

Q38:

This is simple mathematical concept

```
long long int n, a, k;
cin >> n >> a >> k;
long long int num = (n-2)*180;
num*=2;
long long int denom = n;
num -= (2*a*denom);
denom*=(n-1);
long long int g = __gcd(num,denom);
num/=g;
denom/=g;
long long int ans = a*denom;
ans = ans + (k-1)*num;
g = __gcd(ans,denom);
cout << ans/g << " " << denom/g << "\n";
```

Q39:

Ask me in class if you explanation of the code

```
string a,b;
cin >> a >> b;
int match = 0,O = 0,B = 0;
for(int i = 0; i < 3; ++i){
    if((a[i]=='o'&&b[i]=='b')||(a[i]=='b'&&b[i]=='o'))match++;
    else if(a[i] == 'b' || b[i] == 'b'){
        B++;
    }
    else if(a[i] == 'o' || b[i] == 'o'){
        O++;
    }
}
if(match+O+B == 3 && O<2 && B!=3){
    cout << "yes\n";
}else cout << "no\n";
```

Q40:

```
int n;
cin >> n;
ll *arr = new ll[n];
loop(i,0,n)cin >> arr[i];
sort(arr,arr+n);
ll ans = LLONG_MAX;
for(int i = 0; i+1 < n; ++i)ans=min(ans,arr[i+1]-arr[i]);
cout << ans << endl;
```

Q41:

```
long long int a,b,c,x,y;
cin >> a >> b >> c >> x >> y;
if(abs(x-b) + abs(y-c) == a && x >= b && y >= c){
    cout << "YES\n";
    return;
}
if(abs(x-c) + abs(y-b) == a && x >= c && y >= b){
    cout << "YES\n";
    return;
}
if(abs(x-b) + abs(y-a) == c && x >= b && y >= a){
    cout << "YES\n";
    return;
}
if(abs(x-a) + abs(y-b) == c && x >= a && y >= b){
    cout << "YES\n";
    return;
}if(abs(x-a) + abs(y-c) == b && x >= a && y >= c){
    cout << "YES\n";
    return;
}
if(abs(x-c) + abs(y-a) == b && x >= c && y >= a){
    cout << "YES\n";
    return;
}
cout << "NO\n";
```