Explanation done in meeting 1-

How to take infinite input

<u>If we don't know how many integers are given for input</u>

```
int x;
while(cin>>x)
{

}
```

<u>If it is given that -1 is present at the last of input</u>

```
while(true)
{
    int x;
    cin>>x;

    if(x==-1)
        break;

}
```

Fast I/O

```
ios_base::sync_with_stdio(0);
cin.tie(0);
cout.tie(0);
```

Prime number

<u>Segmented sieve</u>

Given l, r as ranges and we have to find the prime number that lie between l and r
max value of r = 10^12
r-l <= 10^6

n= r-l+1;
a[n];
l, l+1, l+2, .. , r
0, 1, 2, , n-1

Code-

```
for(int i=0;i<n;i++)
{
	x = i+l;
	for(int j=0;j<v.size();j++)
	{
		if(x%v[j] ==0)
		{
		    f=0;
		    break;
		}
	}
}
```

Bitwise operators

Consider 2 integers a and b
a= 9 1001
B = 14 1110

or (|) A|b = 1111 15
And (&) a&b = 1000 8
Not (~) A = 0110 6
Xor (^) a^b = 0111 7
<< (*2) a<<1 10010
>> (/2) a>>1 0100

Some properties of these operators

1 | n = 1
0 | n = n
n | n = n
0 & n = 0
1 & n = n
n & n = n
1 ^ n = ~n
0 ^ n = ~n
n ^ n = 0

How to find whether a number is power of 2 or not

```
while(n%2 ==0)
{
        N = n/2;
}

if(n==1)
        Return true;
Else
        Return false;
```

<u>If we have to find this in O(1) constant time</u>

N-1 0111111
N 1000000

N&(N-1) 0000000
N^(N-1) 1111111 problem is that we also have to find this number

n&(n-1) ==0
n^(n-1) == ()

<u>To count number of set bits</u>

cout<<__builtin_popcount(n);
Read more builtin function from gfg

<u>Parity</u> - count number of set bits in a number and find it's parity accordingly

Even = 0
Odd = 1

<u>Some STL functions used in number theory</u>

min(a,b)
min(a,min(b,c))
max(a,b)
__gcd(a,b)
LCM = a*b/__gcd(a,b)

Explanation (Meeting 2) -

- Print pascal triangle till n rows. For n=5 given below

```
   1
  11
 121
1331
14641
```

```
0C0     1
1C0 1C1         1 1
2C0 2C1 2C2             1 2 1
```

Code:

```
vector<int> a;
a.push_back( 1);
a.push_back(1);

for(int i=2;i<=n;i++)
{
        // print spaces according to row number

        vector<int> b;
        b.push_back(1);
        cout<<1<<" ";
        for(int j=0;j<a.size()-1;j++)
        {
                b.push_back(a[i]+a[i+1]);
                cout<<a[i]+a[i+1]<<" ";
        }
        b.push_back(1);
        cou<<1<<" \n";
        A = b;
}
```

Modulus operator : %

a%b = 5%3 = 2
1%5 = 1
(-a)%b = (b-a)%b = -2%5 = 3%5 = 3

(a+b)%m = (a%m + b%m)%m

(a-b)%m = (a%m - b%m +m)%m
(a*b)%m = ((a%m)*(b%m))%m
(a/b)%m = ((a%m)*(b^-1 %m))%m

Read about modulo inverse (euclid - extended euclid)

Fibonacci - 0 1 1 2 3 5 8 13
f(n) = f(n-1) + f(n-2)

4 methods of finding nth fibonacci number
Recursive - O(2^n) O(n)(stack space)
Iterative - O(n) DP - O(n), else O(1)
Matrix - O(log n) O(log n)
Binet’s formula - O(1) O(1)

Matrix

$$\begin{matrix}1 & 1\\ 1 & 0\end{matrix} \quad \begin{matrix}1 & 1\\ 1 & 0\end{matrix} \quad = \begin{matrix}2 & 1\\ 1 & 1\end{matrix} \quad \begin{matrix}1 & 1\\ 1 & 0\end{matrix} \quad = \begin{matrix}3 & 2\\ 2 & 1\end{matrix}$$

Try to code it (refer gfg)

Binet’s formula -

Fn = {[(√5 + 1)/2] ^ n} / √5

Tribonacci -- 0 0 1 1 2 4 7

Catalan number = 2nCn / (n+1)

(n+1) = (0)(n) + (1)(n-1) + (2)(n-2) + … (n)(0)

N , 5
floor()
(n/5) + (n/25) + (n/125) ---

Fast expo
a^b
B even a^(b/2) * a^(b/2)

B odd a^(b/2) * a^(b/2) * a

a=5, b=7
5^7 = 5^(3) * 5^(3) * 5
5^3 = 5^(1) * 5^(1) * 5

A = a%m
B = b%m

5^7 = 5^(3) * 5^(3) * 5
5^3 = (5^(1)%m * 5^(1)%m * 5)%m