- **(A / B) = ((A % Mod) * (BinExp(B , Mod-2) % Mod)) % Mod;**

Bits:

- **Bitwise NOT(~):** inverts all bits of it. [a = 1001_2 **-> (~a) = 0110**]
- (**N / 2) == (N >> 1);** (**N * 2**) == (**N << 1);**
- (**2^N**) **== (1**LL **<< N);** **=> N = (1LL <<** (long long)**log2**(N) **);**
- **is_power_of_two(val)** => **(val & (val - 1)) == 0;**
- **CheckBit(val, pos) => (val & (1LL << pos));**
- **SetBit(val, pos) => (val |= (1LL << pos));**
- **ClearBit(val, pos) => (val &= ~(1LL << pos));**
- **FlipBit(val, pos) => (val ^= ~(1 << pos));**
- **MSB(mask) => 63 - __builtin_clzll(mask);** [Most Significant Bit position]
- **LSB(mask) => __builtin_ctzll(mask);** [Least Significant Bit position]
- **__builtin_popcount(**x**):** This function is used to count the number of one's(set bits) in an integer(32 bits). Similarly you can use **__builtin_popcountll(**x**)** for **long long** data types (64 bits). Ex: x = **5** (101) => ans=2 ;

Bitset Function:

bitset< highest_Bit_number **> name(data);**

- **bitset<**64**>** b1(val); or**, bitset<**4**>**b2("1011"); => auto-convert to binary;
- **to_ulong():** Converts the contents of the **bitset** to an **unsigned long integer**; [Ex: b1 = 1001, int val = b1.**to_ulong**(); => val = 9;]
- **to_string():** Converts the contents of the **bitset** to a **string**; [Ex: b1 = 1001, s1 = b1.to_string(); => s1= "1001";]
- **count():** returns the total number of **set bits**(1); [Ex: b1=1001; bit= b1.count(); => bit =2;]

Combination(C):

- If, **Order Doesn't Matter** and **Repetition Allowed** then, Possibilities, $^{n}C_{r} = \frac{n!}{r!(n-r)!}$
- If, **Order Doesn't Matter** and **Repetition Not Allowed** then, Possibilities, $^{n}C_{r} = \frac{(n+r-1)!}{r!(n-1)!}$

Permutation(P):

- If, **Order Matter** and **Repetition Allowed** then, Possibilities = n^r
- If, **Order Matter** and **Repetition Not Allowed** then, Possibilities = $\frac{n!}{(n-r)!}$