Q1. Fibonacci Number The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1.

```
class Solution {
    public int fib(int n) {
         int n1=0,n2=1;
        if(n==0)
        {
            return 0;
        }
        else if(n==1)
        {
            return 1;
        }
        return fib(n-1)+fib(n-2);
    }
}
```

Q2. Power of Two Given an integer n, return true if it is a power of two. Otherwise, return false. An integer n is a power of two, if there exists an integer x such that n == 2x

```
class Solution {
    public boolean isPowerOfTwo(int n) {
        if(n == 1) return true;
        if(n == 0) return false;
        return n%2 == 0 ? isPowerOfTwo(n/2) : false;
    }
}
```

Q3. Pow(x, n) Implement pow(x, n), which calculates x raised to the power n (i.e., xn).

```
class Solution {
    public double myPow(double x, int n) {
        return Math.pow(x,n);
    }
}
```

Q4. Elimination Game Starting from left to right, remove the first number and every other number afterward until you reach the end of the list. Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers. Keep repeating the steps again, alternating left to right and right to left, until a single number remains. You have a list arr of all integers in the range [1, n] sorted in a strictly increasing order. Apply the following algorithm on arr: Given the integer n, return the last number that remains in arr.

```
class Solution {
    public int lastRemaining(int n) {
        if(n==1)
        {
            return 1;
        }
        return 2*(1+n/2-lastRemaining(n/2));
    }
}
```

Q5. K-th Symbol in Grammar We build a table of n rows (1-indexed). We start by writing 0 in the 1st row. Now in every subsequent row, we look at the previous row and replace each occurrence of 0 with 01, and each occurrence of 1 with 10.

```
class Solution {
    public int kthGrammar(int n, int k) {
        if(n < 3)
            return k - 1;
        int half = (int)Math.pow(2, n - 2);
        if(k <= half)
            return kthGrammar(n - 1, k);
        return kthGrammar(n - 1, k - half) == 0? 1 : 0;
    }
}
```

Q6. Count Good Numbers A digit string is good if the digits (0-indexed) at even indices are even and the digits at odd indices are prime (2, 3, 5, or 7).

```
class Solution {
    public long mod = 1000000007;
    public int countGoodNumbers(long n) {
        long odd = n / 2;
        long even = (n + 1) / 2;
        return (int)((((pow(5, even)) % mod) * ((pow(4, odd)) % mod)) % mod);
    }
    public long pow(long x, long n) {
        if(n == 0) return 1;

        long temp = pow(x, n/2);

        if(n % 2 == 0) return (temp * temp) % mod;
        else return (temp * temp * x) % mod;
    }
}
```

Q7. Permutation Sequence "123" "132" "213" "231" "312" "321" The set [1, 2, 3, ..., n] contains a total of n! unique permutations. By listing and labeling all of the permutations in order, we get the following sequence for n = 3: 1. 2. 3. 4. 5. 6. Given n and k, return the kth permutation sequence.

```
class Solution {
    public String getPermutation(int n, int k) {
        List<Integer>list=new ArrayList<>();
        int fact=1;
        for(int i=1;i<n;i++){
            fact=fact*i;
            list.add(i);
        }
        list.add(n);
        k=k-1;
        String ans="";
        while(true){
            ans+=list.get(k/fact);
            list.remove(k/fact);
            if(list.size()==0){
```

```
                break;
            }
            k=k%fact;
            fact=fact/list.size();
        }
        return ans;
    }
}
```

Q8. Power of Four Given an integer n, return true if it is a power of four. Otherwise, return false. An integer n is a power of four, if there exists an integer x such that n == 4x.

```
class Solution {
    public boolean isPowerOfFour(int n) {
        if(n == 1) return true;
        if(n == 0) return false;
        return n%4 == 0 ? isPowerOfFour(n/4) : false;
    }
}
```