DP
Lesson 2

1. Count ways to reach the n'th stair
2. reach-the-nth-point
3. Count-number-of-hops
4. Count-ways-to-express-n-as-the-sum-of-13-and-4
5. Count numbers containing 4
6. compute-sum-of-digits-in-all-numbers-from-1-to-n
7. Count-the-number-of-ways-to-tile-the-floor-of-size-n-x-m-using-1-x-m-size-tiles
8. Telephone Number or Involution Number
9. Find-optimum-operation
10. Minimum-steps-to-minimize-n-as-per-given-condition
11. High-effort-vs-low-effort
12. sum-of-all-substrings-of-a-number
13. Max-rope-cutting
14. Friends-pairing-problem
15. Letter-writer

Friends Pairing Problem

Given N friends, each one can remain single or can be paired up with some other friend. Each friend can be paired only once. Find out the total number of ways in which friends can remain single or can be paired up.

Note: Since answer can be very large, return your answer mod 10^9+7.

Example 1:

Input:N = 3

Output: 4

Explanation:

{1}, {2}, {3} : All single

{1}, {2,3} : 2 and 3 paired but 1 is single.

{1,2}, {3} : 1 and 2 are paired but 3 is single.

{1,3}, {2} : 1 and 3 are paired but 2 is single.

Note that {1,2} and {2,1} are considered same.

Example 2:

Input: N = 2

Output: 2

Explanation:

{1} , {2} : All single.

{1,2} : 1 and 2 are paired.

```
#define MOD 1000000007
vector<int>ans;
int frnd(int n)
{
   if(n==1)
      return 1;
   if(n==2)
      return 2;
   if(ans[n]==-1)
   {
      long long x=frnd(n-1)%MOD;
      long long y=(((long long)(n-1))%MOD)*((long long)frnd(n-2))%MOD;
      ans[n]=(int)((x+y)%MOD);
   }
   return ans[n];
}
class Solution
{
```

```
public:
    int countFriendsPairings(int n)
    {
        // code here
        ans.assign(n+1,-1);
        return frnd(n);
    }
};
```

```
class Solution
{
public:
    int countFriendsPairings(int n)
    {
        // code here
        int p=1;
        int q=2;
        int k;
        if(n==1)    return p;
        if(n==2)    return q;
        for(long long i=3; i<=n; i++)
        {
            k=(q+(((i-1)%MOD)*p))%MOD;
            p=q;
            q=k;
        }
        return k;
    }
};
```

Letter Writer

Geek works at the post office and he writes the following types of letters.

Corporate Letters: 12 letters of this type can be written in an hour.

Informal Letters: 10 letters of this type can be written in an hour.Given N number of letters, find the minimum number of hours he needs to generate a combination of both types without wasting any time.

```
1.  int minHours(int N){
2.          int dp[N+1];
3.        if(N<10)
4.           return -1;
5.        dp[0]=0;
6.        for(int i=1; i<10; i++)
7.           dp[i]=-1;
8.        dp[10]=1;
9.        if(N==10)
10.          return 1;
11.       dp[11]=-1;
12.       if(N==11)
13.          return -1;
14.       dp[12]=1;
15.       for(int i=13; i<=N; i++)
16.       {
17.          dp[i]=INT_MAX;
18.          if(dp[i-10]!=-1)
19.             dp[i]=min(dp[i],dp[i-10]+1);
20.          if(dp[i-12]!=-1)
21.             dp[i]=min(dp[i],dp[i-12]+1);
22.          if(dp[i]==INT_MAX)
23.             dp[i]=-1;
24.       }
25.       return dp[N];   }
```

Count numbers containing 4

Count the numbers between 1 to **N** containing 4 as a digit.

Input:

N = 9

Output:

1

Explanation:

4 is the only number between 1 to 9

which contains 4 as a digit.

```
bool has4(int x)
{
   while (x != 0)
   {
      if (x%10 == 4)
         return true;
      x   = x /10;
   }
   return false;
}
   int countNumberswith4(int n) {
      // code here
      int result = 0;
      for (int x=1; x<=n; x++)
          result += has4(x)? 1 : 0;
   return result;
   }
```

Compute sum of digits in all numbers from 1 to n

Given a number **N**, find the total sum of digits of all the numbers from **1** to **N**.

Example 1:

Input:

N = 5

Output:

15

Explanation:

Sum of digits of number from 1 to 5:

1 + 2 + 3 + 4 + 5 = 15

Example 2:

Input:

N = 12

Output

51

Explanation:

Sum of all digits from 1 to 12 is:

1+2+3+4+5+6+7+8+9+(1+0)+(1+1)+(1+2) = 51

```
int Sum(int N, int digits[])
{
   if(N<10)
      return  N*(N+1)/2;
   int d=log10(N);
   int x=N/(int)pow(10,d);
   int y=N%((int)pow(10,d));
   return (x*digits[d]+(x*(x-1)/2)*pow(10,d))+x+x*y+Sum(y,digits);
}
class Solution{
public:
   int sumOfDigits(int N){
      //code here
      int d=log10(N);
      int digits[d+1];
      digits[0]=0;
      for(int i=1; i<=d; i++)
      {
         digits[i]=10*digits[i-1]+45*pow(10,i-1);
      }
      return Sum(N,digits);
   }
};
```

Count the number of ways to tile the floor of size n x m using 1 x m size tiles

Given a floor of size n x m and tiles of size 1 x m. The problem is to count the number of ways to tile the given floor using 1 x m tiles. A tile can either be placed horizontally or vertically.Both n and m are positive integers and 2 < = m.

Example 1 :

Input: n = 2, m = 3

Output: 1

Explanation: There is only one way to tile the given floor.

Example 2 :

Input: n = 4, m = 4

Output: 2

Explanation: There are two ways to tile the given floor.One way is to place 1 x 4 size of tile vertically and another one is to place them horizontally.

```
1.  #define MOD 1000000007
2.  class Solution{
3.          public:
4.                  int countWays(int n, int m)
5.          {
6.             int dp[n+1];
7.             for(int i=0; i<m && i<=n; i++)
8.                dp[i]=1;
9.             for(int i=m; i<=n; i++)
10.               dp[i]=(dp[i-1]+dp[i-m])%MOD;
11.            return dp[n];
12.         }
13. };
```

Sum of all substrings of a number

Given an integer S represented as a string, the task is to get the sum of all possible sub-strings of this string.

As the answer will be large, print it modulo 10^9+7.

Example 1:

Input:

S = 1234

Output: 1670

Explanation: Sum = 1 + 2 + 3 + 4 + 12 +

23 + 34 + 123 + 234 + 1234 = 1670

Example 2:

Input:

S = 421

Output: 491

Explanation: Sum = 4 + 2 + 1 + 42 + 21

```
class Solution
{
  public:
  //Function to find sum of all possible substrings of the given string.
  long long sumSubstrings(string s){

      // your code here
      long long prev=s[0]-'0';
      long long ans=prev;
      long long l=s.length();
      for(int i=1; i<l; i++)
      {
         prev=(((i+1)*(s[i]-'0'))%MOD+prev*10)%MOD;
         ans=(ans+prev)%MOD;
      }
      return ans;
  }
};
```