Problem 1
You are given a list of positive integers in an array having no duplicate elements. Find all the pairs of integers having a difference at most one.

Input:
```
Arr = [57, 2, 1, 4, 89, 7, 9, 10, 34, 8, 56, 11, 90, 6]
```

Output:
```
(2, 1) (7, 8) (7, 6) (9, 10) (9, 8) (10, 11) (57, 56) (89, 90)
```

Note: There is no restriction on the output order.

First solution

```
Def findDiffWith1( arr ):
        Result = []
        N = len( arr )
        For i in range( n ):
           For j  in range( i+1, n ):
                if( Math.abs( a[i] - a[ j ] ) == 1 ):
                        result.append( (a[i], a[j] )


Return result
```

-> O(n^2)

Second solution

```
Def findDiffWith1( arr ):
        Result = []
        sort( arr )  # O( n log n )

        # 1 , 2, 4, 6, 7, 8, 9, 10, 11, 34, 57, 89, 90

        N = len( arr  )

        For i in range( n - 1 ): # O( n )
                if( ar[ i+1 ] - ar[ i ] == 1 ):
                        result.append( (a[i], a[j] ) )

        Return result
```

-> O(n log n)

Third solution

```
def findDiffWith1(arr):
        mp = {}
```

```
Result = []
for i in arr:  # O(n)
        mp[i] = 1


# Arr = [57, 56, 2, 1, 4, 89, 7, 9, 10, 34, 8, 11, 90, 6]
          1,  1,    1
For k in mp:
        if( mp.get( k+1) != None ):
                result.append( k, K+1 )
Return result
```

-> O(n)

Problem 2:

You are climbing a staircase. Each time you can either climb 1 or 2 steps. If I give you the number of steps in the staircase, can you tell me how many ways you can reach the top?

```
Input: 2
Output: 2
Explanation:
1 step + 1 step
2 step

Input: 3
Output: 3
Explanation:
1 step + 2 step
2 step + 1 step
1 step + 1 step + 1 step

Input: 44
Output: 1134903170
```

```
numSteps[ 1 ] = 1
numSteps[ 2 ] = 2

numSteps[ 3 ] = numSteps[ 2 ] + numSteps[ 1 ]
```

```
numSteps[ n ] = numSteps[ n - 1 ] + numSteps[ n - 2 ]
```

```
def getSteps(n):
     if( n == 1 ):
          Return 1
     If( n == 2 ):
          Return 2
     Return getSteps( n - 1 ) + getSteps( n - 2 )
```

```
Time : O( 2^n ) # Time complexity
Space : O( 2^n )
```

```
def getSteps(n):
     if( n < 1 ):
          Return None
     numSteps = {}
     numSteps[ 1 ] = 1
     numSteps[ 2 ] = 2
     if( n < 3 ):
          Return numSteps[ n ]

     For i in range( 3, n + 1 ):
          numSteps[ i ] += numSteps( i - 1 ) + numSteps( i - 2 )
     Return numSteps[ n ]
```

```
Time : O( n ) # Time complexity
Space : O( n )
```

```
def getSteps(n):
     if( n < 1 ):
          Return None
     numStepsA = 1
```

```
    numStepsB = 2
    if( n < 3 ):
        Return numSteps[ n ]

    For i in range( 3, n + 1 ):
        numStepsC += numStepsA + numStepsB
        numStepsA = numStepsB
        numStepsB = numStepsC
    Return numSteps[ n ]


Time : O( n ) # Time complexity
Space : O( n )
```