DAY 3 Q1

```
def maxProfit(price, n):
   profit = [0]*n
   max_price = price[n-1]

   for i in range(n-2, 0, -1):

      if price[i] > max_price:
         max_price = price[i]
      profit[i] = max(profit[i+1], max_price - price[i])
   min_price = price[0]

   for i in range(1, n):

      if price[i] < min_price:
         min_price = price[i]
      profit[i] = max(profit[i-1], profit[i]+(price[i]-min_price))

   result = profit[n-1]

   return result
price = [7,1,5,3,6,4]
print ("Maximum profit is", maxProfit(price, len(price)))
```

DAY 3 Q3

```
def numIdenticalPairs(self, nums: List[int]) -> int:
    return sum((n)*(n-1)//2 for n in Counter(nums).values())
```

DAY 3 Q4

```
def add_binary_nums(x, y):
    max_len = max(len(x), len(y))

    x = x.zfill(max_len)
    y = y.zfill(max_len)

    # initialize the result
    result = ''

    # initialize the carry
    carry = 0

    # Traverse the string
    for i in range(max_len - 1, -1, -1):
      r = carry
      r += 1 if x[i] == '1' else 0
      r += 1 if y[i] == '1' else 0
      result = ('1' if r % 2 == 1 else '0') + result
      carry = 0 if r < 2 else 1     # Compute the carry.

    if carry !=0 : result = '1' + result

    return result.zfill(max_len)
```

DAY 3 Q6

```
a=int(input())
p=0
b=a
while(b>0):
   r=b % 10
   p=p*10+r
   b=b//10
if (p==a):
   print ("Mirror")
else:
   print("No Mirror")
```

DAY 3 Q8

```
class Solve():
   def Anagrams( self, li ):
       dictionary = {}
       for word in li:
           sortedWord = ''.join(sorted(word))
           if sortedWord not in dictionary:
               dictionary[sortedWord] = [word]
           else:
               dictionary[sortedWord] += [word]
       return [dictionary[i] for i in dictionary]


if __name__ == '__main__':
    li=[]
    n=int(input("enter number of words"))
    for i in range(1,n+1,1):
        ele=input("enter")
        li.append(ele)
    print(li)
print(Solve().Anagrams(li))
```

DAY 3 Q10

```
def editDistance(str1, str2, m, n):
    if m == 0:
        return n
    if n == 0:
        return m
    if str1[m-1] == str2[n-1]:
        return editDistance(str1, str2, m-1, n-1)
    return 1 + min(editDistance(str1, str2, m, n-1),
                   editDistance(str1, str2, m-1, n),
                   editDistance(str1, str2, m-1, n-1))
str1 = input("enter string1")
str2 = input("enter string2")
print (editDistance(str1, str2, len(str1), len(str2)))
```