1) **Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.**
An input string is valid if:
Open brackets must be closed by the same type of brackets.
Open brackets must be closed in the correct order.
Example 1: Input: s = "()"
Output: true

Example 2: Input: s = "()[]{}"
Output: true

Example 3: Input: s = "(]"
Output: false

Example 4: Input: s = "([)]"
Output: false

Example 5: Input: s = "{[]}"
Output: true

Constraints:

1 <= s.length <= 104 **S consists of parentheses only '()[]{}'.**

Code:

```
d = {'(':')', '{':'}','[':']'}
s=input("Enter string:")
stack = []
for i in s:
    if i in d:
        stack.append(i)
    elif len(stack) == 0 or d[stack.pop()] != i:
        print("False")
        break
if (len(stack) == 0):
    print("True")
else:
    print("False")
```

2)Best Time to Buy and Sell Stock.

You are given an array prices where prices[i] is the price of a given stock on the ith day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Example 1:

Input: prices =[7,1,5,3,6,4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5. Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Example2:

Input: prices = [7,6,4,3,1]

Output: 0

Explanation: In this case, no transactions are done and the max profit = 0.

Code:

```
prices = []
n= int(input("Enter number of elements : "))
for i in range(0, n):
    ele = int(input())
    prices.append(ele)
if len(prices) == 0:
    print(0)


max = prices[len(prices)-1]
profit = 0


for item in prices[::-1]:
    if max – item > profit:
```

```
        profit = max – item
    if item > max:
        max = item
```

3)Merge Intervals

Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.

Example 1:

Input: intervals=[[1,3],[2,6],[8,10],[15,18]]

Output: [[1,6],[8,10],[15,18]]

Explanation: Since intervals [1,3] and [2,6] overlaps, merge them into [1,6].

Example 2:

Input: intervals = [[1,4],[4,5]]

Output: [[1,5]]

Explanation: Intervals [1,4] and [4,5] are considered overlapping.

Code:

```
res=[]
intervals=[]
sublist=[]
n=int(input("enter number of intervals:"))
for i in range(n):
    for j in range(2):
        sublist.append(int(input()))
    intervals.append(sublist)
    sublist=[]
for beg, end in sorted(intervals):
    If not res or res[-1][1] < beg:
        res += [[beg, end]]
    else:
```

```
res[-1][1] = max(res[-1][1], end)
```