This problem is the roman to integer problem:

```
class Solution:
    def romanToInt(self, s: str) -> int:

        roman = {"I" : 1, "V" :5, "X" : 10,
                 "L" : 50, "C" : 100, "D" : 500, "M" : 1000 }
        res = 0

        for i in range(len(s)):
            if i + 1 < len(s) and roman[s[i]] < roman[s[i + 1]]:
                res -= roman[s[i]]
            else:
                res += roman[s[i]]

        return res
```

First off again we will use a hashmap to store the roman numerals data by saying:

```
roman = {"I" : 1, "V" :5, "X" : 10,
                 "L" : 50, "C" : 100, "D" : 500, "M" : 1000 }
```

Because thats the values given to us

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

We will then say res = 0 initially but it was change at the end when we return res

```
 res = 0
```

We will now use a for loop to check the length of s which will be our string also known as our roman numeral and i will be our index.

```
   for i in range(len(s)):
```

Now using our if conditions we will check our hashmaps if i + 1 is in bounds then we will find the value of index i using roman and we will compare the key s to i and check using < roman[s[i + 1]]: if it is smaller and if it is it will be subtracted from the result

so we will do

```
        res -= roman[s[i]]
            else:
                res += roman[s[i]
```

Otherwise we will add

Then we will return the res

We have a problem of “valid parentheses” this problem states that given a string which will be named s that will only contain the values of a parentheses such as “(“ “)” or “[“ “]” or “{“ “}”

Solution:

```
stack = []

brackets = {")" : "(" , "}" : "{", "]" : "["}

for i  in s:
    if i in brackets:
        if stack and stack[-1] == brackets[i]:
            stack.pop()
        else:
            return False
    else:
        stack.append(i)
return True if not stack else False
```

We need to make sure that the string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

In order to solve this problem we simply need to use a hashmap to compare the two strings to determine if the opening parentheses will match the closing parentheses.

First we will set up our empty stack and then set our values in the hashmap using:

```
stack = []

brackets = {")" : "(" , "}" : "{", "]" : "["}
```

```
Then we will loop through using a for loop

for c  in s:

We will use i variable this time for our index in s:
```

```
for c  in s:
```

Now we will set up conditions to compare the parentheses.

we will say if the i is in the bracket and if stack and stack -1. -1 will check the last value in the stack and then we will compare it using == to the bracket hashmap we set up and if they DO MATCH then we can pop from our stack which means simply to remove it from the stack and when we return an empty stack the code will return true.

```
if i in brackets:
                if stack and stack[-1] == brackets[i]:
                    stack.pop()
```

NOW IF it is not true that the stack and bracket hashmap match we will use an else statement to return false:

```
  else:
                    return False
```

We can only return true if our stack is empty and we use this code to confirm the stack will return true

```
            else:
                stack.append(i)
        return True if not stack else False
```