

WEEK-2

Leetcode Bootcamp

Q1.

Code:

```
class Solution(object):
    def myAtoi(self, s):
        s = s.strip()
        if not s:
            return 0
        i = 0
        sign = 1
        result = 0
        if s[i] == '-' or s[i] == '+':
            sign = -1 if s[i] == '-' else 1
            i += 1
        while i < len(s) and s[i].isdigit():
            digit = int(s[i])
            if result > (2**31 - 1 - digit) // 10:
                return 2**31 - 1 if sign == 1 else -2**31
            result = result * 10 + digit
            i += 1
        return sign * result
```

The screenshot displays the LeetCode interface for the 'String to Integer (atoi)' problem. The top navigation bar includes 'Problem List', 'Run', 'Submit', and 'Premium'. The problem status is 'Accepted' with 1095/1095 testcases passed. The user 'rushabh910' submitted the solution on Mar 11, 2025, at 22:37. The runtime performance is shown as 0 ms, beating 100.00% of other solutions, with a memory usage of 17.70 MB, beating 77.06%. A bar chart visualizes the runtime performance across different time intervals. The code editor shows the Python3 solution, which is identical to the code provided in the previous block. The test case '42' is shown as accepted, with a runtime of 0 ms.

```
class Solution(object):
    def myAtoi(self, s):
        s = s.strip()
        if not s:
            return 0
        i = 0
        sign = 1
        result = 0
        if s[i] == '-' or s[i] == '+':
            sign = -1 if s[i] == '-' else 1
            i += 1
        while i < len(s) and s[i].isdigit():
            digit = int(s[i])
            if result > (2**31 - 1 - digit) // 10:
                return 2**31 - 1 if sign == 1 else -2**31
            result = result * 10 + digit
            i += 1
        return sign * result
```

Q2.

Code:

```

class Solution:
    def findAnagrams(self, s: str, p: str) -> List[int]:
        list = []
        freqS = [0] * 26
        freqP = [0] * 26

        if len(s) < len(p):
            return list

        for i in range(len(p)):
            freqS[ord(s[i]) - ord('a')] += 1
            freqP[ord(p[i]) - ord('a')] += 1

        start = 0
        end = len(p)

        if freqS == freqP:
            list.append(start)

        while end < len(s):
            freqS[ord(s[start]) - ord('a')] -= 1
            freqS[ord(s[end]) - ord('a')] += 1

            if freqS == freqP:
                list.append(start + 1)

            start += 1
            end += 1

        return list

```

The screenshot shows a web browser with the URL <https://leetcode.com/problems/find-all-anagrams-in-a-string/>. The page displays the problem description, a submission status of 'Accepted', and performance metrics: 25 ms runtime (beats 94.33%) and 18.54 MB memory (beats 16.20%). The code is shown in a dark-themed editor, and the test results section shows the input 's = "cbaebabacd"' and the output '["0", "1", "6", "7"]'.

Q3.

Code:

```
class Solution(object):  
    def reverseWords(self, s):  
        words = s.split()  
        reversed_words = words[::-1]  
        reversed_string = ' '.join(reversed_words)  
  
        return reversed_string
```

