2.Medium\04.Implement_atoi.cpp

```
/*
QUESTION:-
Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi function).

The algorithm for myAtoi(string s) is as follows:

Read in and ignore any leading whitespace.
Check if the next character (if not already at the end of the string) is '-' or '+'. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.
Convert these digits into an integer (i.e. "123" -> 123, "0032" -> 32). If no digits were read, then the integer is 0. Change the sign as necessary (from step 2).
If the integer is out of the 32-bit signed integer range [-231, 231 - 1], then clamp the integer so that it remains in the range. Specifically, integers less than -231 should be clamped to -231, and integers greater than 231 - 1 should be clamped to 231 - 1.
Return the integer as the final result.
Note:

Only the space character ' ' is considered a whitespace character.
Do not ignore any characters other than the leading whitespace or the rest of the string after the digits.


Example 1:

Input: s = "42"
Output: 42
Explanation: The underlined characters are what is read in, the caret is the current reader position.
Step 1: "42" (no characters read because there is no leading whitespace)
         ^
Step 2: "42" (no characters read because there is neither a '-' nor '+')
         ^
Step 3: "42" ("42" is read in)
           ^
The parsed integer is 42.
Since 42 is in the range [-231, 231 - 1], the final result is 42.
Example 2:

Input: s = "   -42"
Output: -42
Explanation:
Step 1: "   -42" (leading whitespace is read and ignored)
            ^
Step 2: "   -42" ('-' is read, so the result should be negative)
             ^
Step 3: "   -42" ("42" is read in)
               ^
The parsed integer is -42.
```

```
Since -42 is in the range [-231, 231 - 1], the final result is -42.

Approach:
1. Initialize an index `i` to track the current position in the string.
2. Skip any leading whitespace by incrementing `i` until a non-whitespace character is encountered.
3. Check if the next character (if not at the end of the string) is `'-'` or `'+'`. Set a `sign` flag accordingly to determine the final result's sign.
4. Read the consecutive digits until a non-digit character is encountered or the end of the input is reached. Convert these digits into an integer.
5. Apply the sign to the integer obtained from the digits.
6. If the integer is out of the 32-bit signed integer range, clamp it to the range [-231, 231 - 1].
7. Return the final integer.

Code:*/
int myAtoi(string s) {
    int i = 0;
    while (i < s.size() && s[i] == ' ')
        i++;
    if (i == s.size())
        return 0;
    bool sign = true; // indicates the number is positive

    if (s[i] == '-') {
        i++;
        sign = false;
    } else if (s[i] == '+')
        i++;

    long long ans = 0;
    while (i < s.size() && (0 <= s[i] - '0' && s[i] - '0' <= 9)) {
        long long digit = s[i] - '0';
        ans = (ans * 10) + digit;
        // handle overflow case
        if (ans > INT_MAX && sign)
            return INT_MAX;
        else if (ans > INT_MAX)
            return INT_MIN;
        i++;
    }
    ans = sign ? ans : -1 * ans;
    return (int) ans;
}

/*
- Time Complexity: The function scans the input string once, resulting in a linear time complexity of O(n), where n is the length of the input string.
- Space Complexity: The function uses a constant amount of extra space, resulting in constant space complexity, O(1).
*/
```