**QPREP7-String to integer atoi**

**Module Introduction**

#### Write a program to take a number string as input and convert it to an integer.

#### Objective

Given an input string representing a number, convert that to an integer value.

Assume the number is within integer range [(-2^31) to (2^31) -1].

If empty input is seen return 0. Handle leading and trailing spaces.

#### Examples

**Example 1**

Input: ' 1003 '

Output: 1003

**Example 2**

Input: '-99'

Output: -99

**Example 3**

Input: '+199'

Output: 199

***SOLUTION STEPS FROM NEXT PAGE:***

**Write down at least 3 examples in the following format. Kindly stick to the format.**

**Suggestion:**

EXAMPLE#1

INPUT:

12344

OUTPUT:

12344

EXAMPLE#2

INPUT:

-98789

OUTPUT:

-98789

EXAMPLE#3

INPUT:

+7658

OUTPUT:

7658

**Detail your problem understanding here**

**Suggestion:**

A string representing a number is provided as input.

It could be a positive or negative number but will be within the range of a 32 bit integer.

This string has to be translated to an integer value and returned.

**Does this problem follow a known algorithmic pattern or standard application of a data structure? If there are multiple approaches, which one would you choose and why? Write down your chosen approach in 2-3 sentences like you would explain to a 10 year old.**

This problem is related to String manipulation and basics of Math. One approach is to loop through the characters, multiply value by 10 and keep adding the next character.

"1234" =>

1\*10 + 2

12\*10 + 3

123\*10 + 4

**Write the pseudocode here in plain English**

preprocessInput(inputString) {

trimSpaces(inputString)

trimPrecedingZeroes(inputString)

trimPrecedingPlusSign(inputString)

isBeginningWithNegativeSign(inputString), mark as a negative number.

If any character other than digits or empty string is seen now, throw error

}

CovertStringToInteger(inputString) {

preprocessInput(inputString)

Initialize Result to 0

Iterate through preprocessedString {

Get integer value of the character

Result = Current Result \* 10 + integer value of current character

}

If number was negative number return -Result

Else return Result

}

**Can you specify a few boundary or edge cases here?**

**Edge cases**

EXAMPLE#2

INPUT:

0

OUTPUT:

0

EXAMPLE#2

INPUT:

(empty)

OUTPUT:

0

EXAMPLE#3

INPUT:

2147483647

OUTPUT:

2147483647

EXAMPLE#4

INPUT:

-2147483648

OUTPUT:

-2147483648

**Write the functions you would create here**

// Return preprocessed string and whether it’s a positive or negative number

<String, Boolean> preprocessInput(inputString)

String trimSpaces(String inputString)

String trimPrecedingZeroes(String inputString)

String trimPrecedingPlusSign(String inputString)

Boolean isBeginningWithNegativeSign(String inputString)

int covertStringToInteger(String input)

#### Summary

Starting with a brief explanation of the problem statement followed by pseudocode and then implementing the solution helps you approach the problem in a systematic way. This methodology helps with easy as well as hard problems.

**Time Complexity: O(n)**

Where n is the length of the input string

**Space Complexity: O(1)**

Since we use constant space independent of the number

#### Concepts

Concepts covered in this Module

* Math
* String

Similar problems

* <https://leetcode.com/problems/reverse-integer/>
* <https://leetcode.com/problems/valid-number/>

SOLUTION:

APPROACH 1:

import java.io.\*;

import java.util.\*;

class StringToIntegerAtoi {

// Implement your solution by completing the below function

public int myAtoi(String str) {

int index = 0;

int total = 0;

int sign = 1;

// Check if empty string

if(str.length() == 0)

return 0;

// remove white spaces from the string

while(index < str.length() && str.charAt(index) == ' ')

index++;

if (index == str.length()) return 0;

// get the sign

if(str.charAt(index) == '+' || str.charAt(index) == '-') {

sign = str.charAt(index) == '+' ? 1 : -1;

index++;

}

// convert to the actual number and make sure it's not overflow

while(index < str.length()) {

int digit = str.charAt(index) - '0';

if(digit < 0 || digit > 9) break;

// check for overflow

if(Integer.MAX\_VALUE / 10 < total || Integer.MAX\_VALUE / 10 == total && Integer.MAX\_VALUE % 10 < digit)

return sign == 1 ? Integer.MAX\_VALUE : Integer.MIN\_VALUE;

total = total\*10 + digit;

index++; // don't forget to increment the counter

}

return total\*sign;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String str = scanner.nextLine();

scanner.close();

int result = new StringToIntegerAtoi().myAtoi(str);

System.out.println(result);

}

}

**Complexity Analysis:**

* **Time Complexity:**
* **Space Complexity:**