REVERSE WORDS IN A STRING (LEETCODE-151)

GitHub: **github.com/BCAPATHSHALA**
Date: 23-12-2023

Ex1

Input

String S = "_A_GOOD___EXAMPLE__"

Output

S = "EXAMPLE_GOOD_A"

Brute force approach:

Step 1: *Trim the input string to remove leading and trailing spaces*

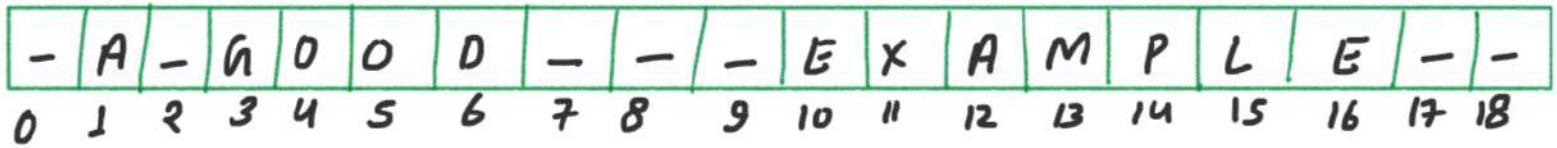
Step 2: *Store all words from string into stack*

Step 3: *Store and delete all elements from stack into string*

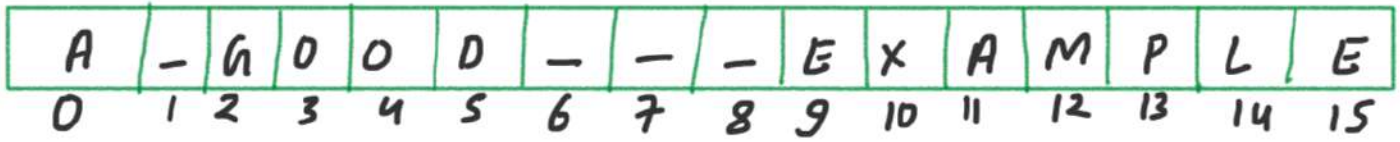
DRY RUN

Ex1

String s = "_A_GOOD___EXAMPLE__"



STEP1 Trim the input string s



Trimmed String

S	A	-	G	O	O	D	-	-	-	E	X	A	M	P	L	E
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	i	i														

STEP 2 STORE EACH WORDS in stack

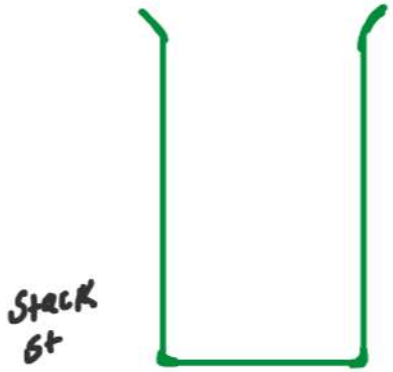
Situation 1

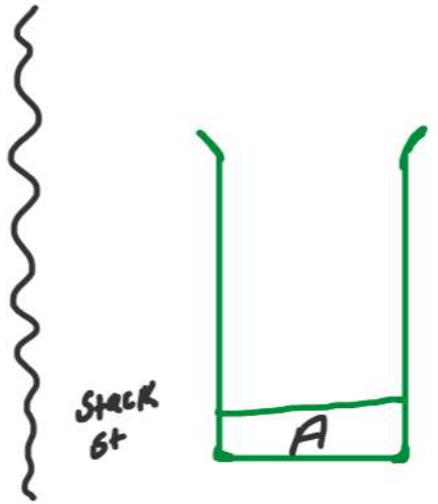
word = ""

if (s[0] != ' ')

↳ word += s[0];

word = "A"





Situation 2

word = ""

if (s[1] == ' ')

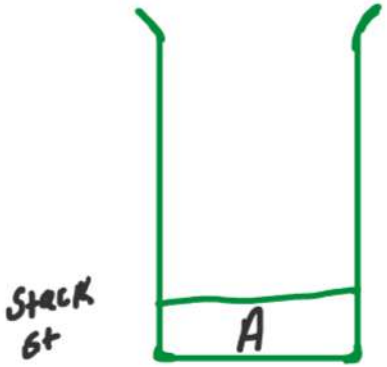
↳ st.push(word);

word = "";

Trimmed String

S	A	-	G	O	O	D	-	-	-	E	X	A	M	P	L	E
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

STEP 2 STORE EACH WORDS in stack



Stack

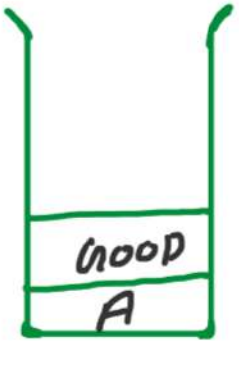
Intuition 3, 4, 5, 6

word = ""

if (s[5] != ' ')

↳ word += s[5];

word = "GOOD"



Stack

Intuition 7

word = "GOOD"

if (s[6] == ' ')

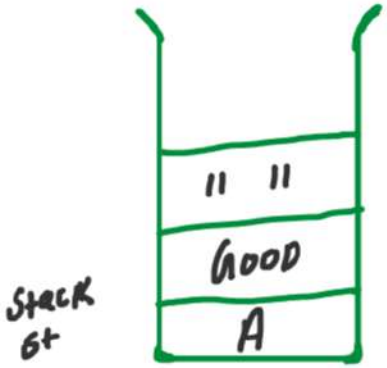
↳ st.push(word);

word = "";

Trimmed String

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	_	G	O	O	D	_	_	_	E	X	A	M	P	L	E

STEP 2 STORE EACH WORDS in stack



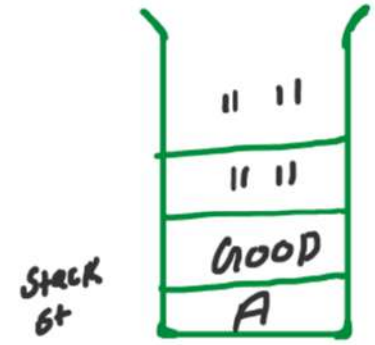
Iteration 8

word = ""

if (s[7] == ' ')

↳ st.push(word);

word = "";



Iteration 9

word = ""

if (s[8] == ' ')

↳ st.push(word);

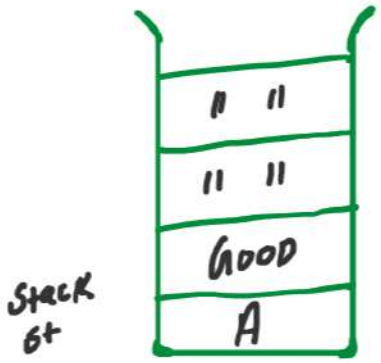
word = "";

Trimmed String

S	A	_	G	O	O	D	_	_	_	E	X	A	M	P	L	E
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

size = 16
i = 16

STEP 2 STORE EACH WORDS in stack

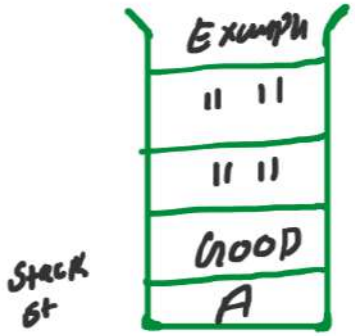


Iteration 10 to 16

word = " "

if (S[15] != ' ')
↳ word += S[15];

word = "EXAMPLE"

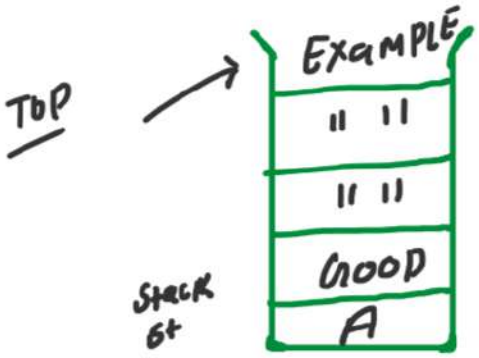


Iteration 17

word = "EXAMPLE"

if (S[17] == ' ' || i == size)
↳ st.push(word);
word = "";

STEP3 Stack to string



Stack size = 5

```
ANS = " "
while ( st.size() != 1) {
  if ( st.TOP().length == 0 )
      ↳ st.POP();

  ELSE
      ↳ Ans += st.TOP() + " ";
        st.POP();
}
stores last TOP elmnt without " " in Ans
  Ans += st.TOP();
```

DRYRUN →

When size = 5 TOP length = 7
ANS = "EXAMPLE_"

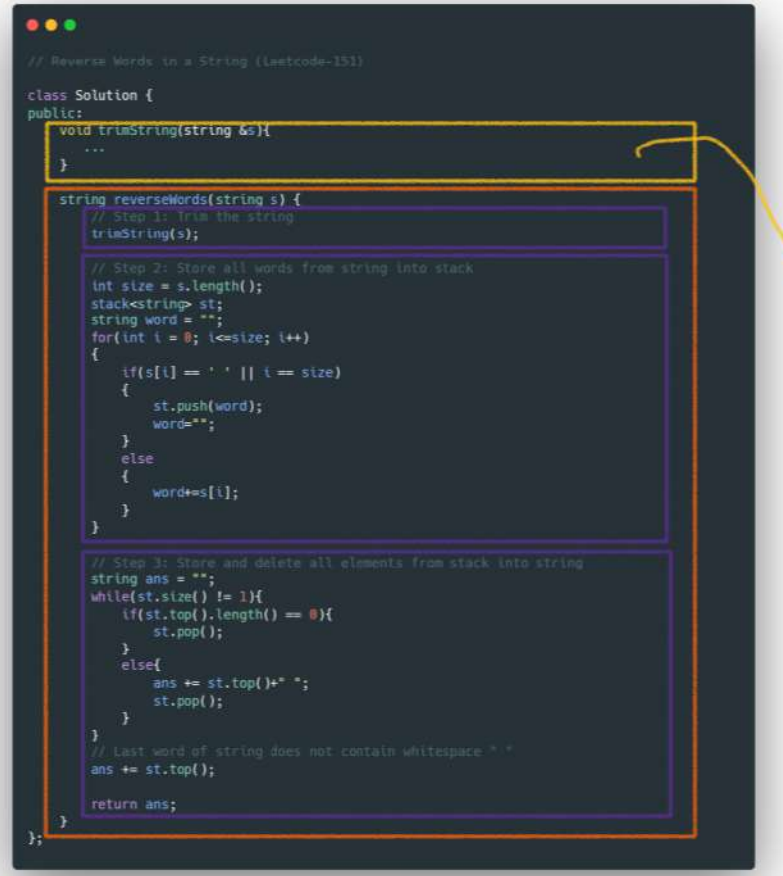
When size = 4 TOP length = 0
ANS = "EXAMPLE_"

When size = 3 TOP length = 0
ANS = "EXAMPLE_"

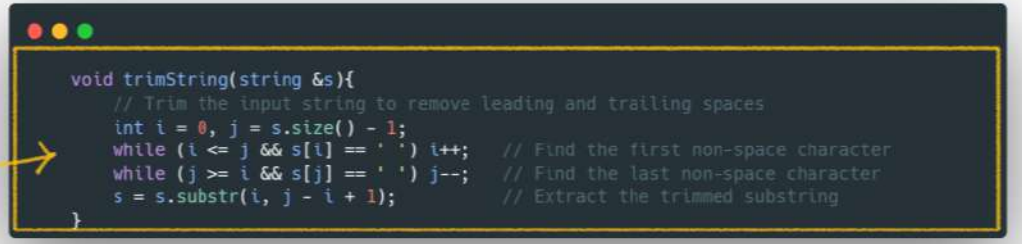
When size = 2 TOP length = 4
ANS = "EXAMPLE_GOOD_"

When size = 1 TOP length = 1
ANS = "EXAMPLE_GOOD_A"

Final output



```cpp
// Reverse Words in a String (Leetcode-151)

class Solution {
public:
    void trimString(string &s){
        ...
    }

    string reverseWords(string s) {
        // Step 1: Trim the string
        trimString(s);

        // Step 2: Store all words from string into stack
        int size = s.length();
        stack<string> st;
        string word = "";
        for(int i = 0; i<=size; i++)
        {
            if(s[i] == ' ' || i == size)
            {
                st.push(word);
                word="";
            }
            else
            {
                word+=s[i];
            }
        }

        // Step 3: Store and delete all elements from stack into string
        string ans = "";
        while(st.size() != 1){
            if(st.top().length() == 0){
                st.pop();
            }
            else{
                ans += st.top()+" ";
                st.pop();
            }
        }
        // Last word of string does not contain whitespace " "
        ans += st.top();

        return ans;
    }
};
```



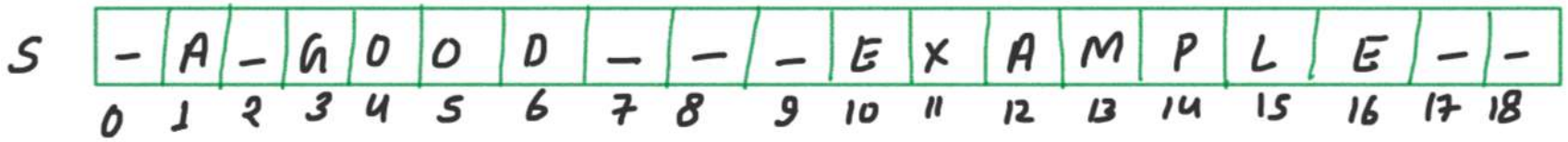
```cpp
void trimString(string &s){
    // Trim the input string to remove leading and trailing spaces
    int i = 0, j = s.size() - 1;
    while (i <= j && s[i] == ' ') i++;   // Find the first non-space character
    while (j >= i && s[j] == ' ') j--;   // Find the last non-space character
    s = s.substr(i, j - i + 1);          // Extract the trimmed substring
}
```

Time complexity: *O(n), traversing the entire string*
Space complexity: *O(n),* ***STACK*** *and* ***ANS*** *variable*

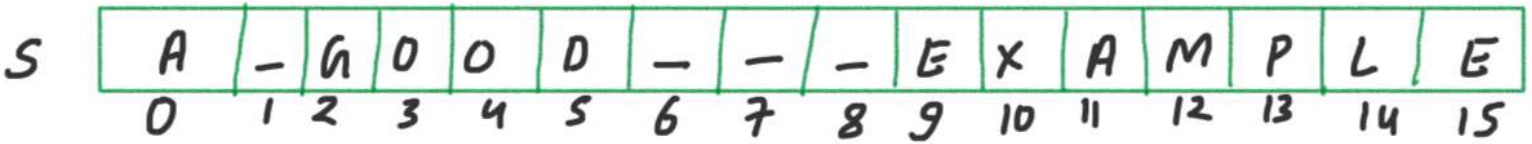
OPTIMAL SOLUTION without taking extra space

Ex1

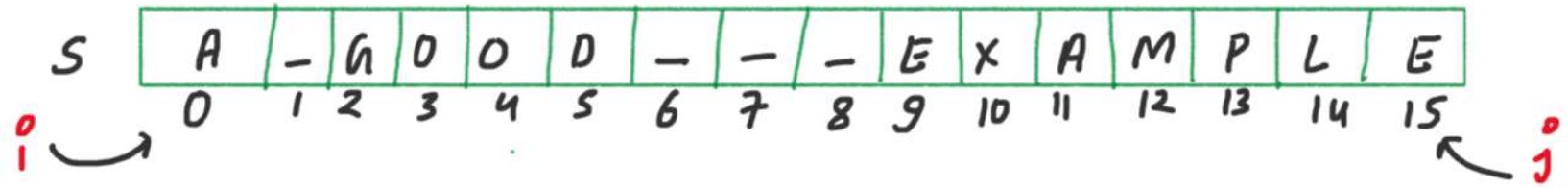
String S = "_A_GOOD___EXAMPLE__"



STEP1 Trim the input string S



STEP2 STORE EACH word in Ans string



word = ""

Ans = ""

ch = s[i]

starting

i=0

if(ch != ' ')

↳ word += s[0];

i++;

word = "A"

Ans = ""

i=1

if(ch == ' ') {

↳ if(word != "") {

↳ if(Ans == "") {

Ans = word;

word = "";

}

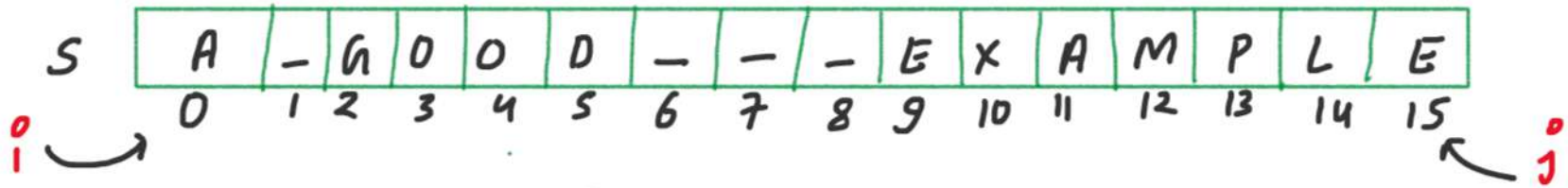
}

i++

we are storing the first word in the Ans

word = ""

Ans = "A"



ch = s[i]

i = 2, 3, 4, 5

```
if(ch != ' ')
    word += s[5];
    i++;
```

word = "GOOD"

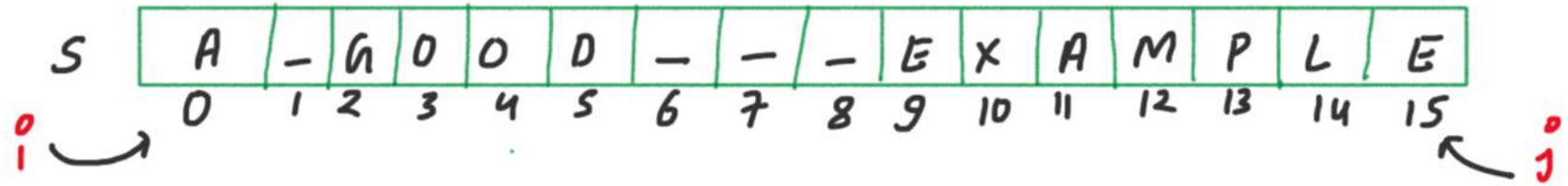
Ans = "A"

i = 6

```
if(ch == ' '){
    if(word != "") {
        if(Ans != "") {
            Ans = word + " " + ANS;
            word = "";
        }
    }
}
i++
```

word = ""

Ans = "GOOD A"



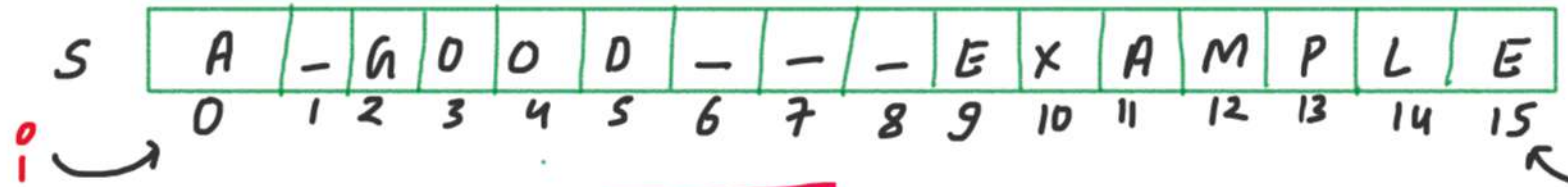
i = 7, 8

```
if( ch == ' ' ){
    if( word == "" ){
        if( Ans != "" ){
            Ans = word + " " + ANS;
            word = "";
        }
    }
}
i++
```

False

ch = s[i]

word = ""
Ans = "GOOD_A"



ch = s[i]

i = 9, 10, 11, 12, 13, 14, 15

```
if(ch != ' ')
    word += s[15];
    i++;
```

word = "EXAMPLE"
Ans = "GOOD_A"

STORE LAST WORD when [i, j)

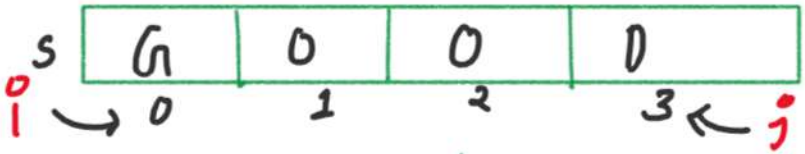
```
if( word != "" ) {
    if( Ans != "" ) {
        Ans = word + " " + Ans;
        word = "";
    }
}
```

word = ""
Ans = "EXAMPLE_GOOD_A"

O/P

EX2

String s = "Good";



word = "hood"

ans = ""

ch = s[i]

i = 0, 1, 2, 3

if(ch != ' ')

↳ word += s[i]

i++

in this call, start First and last Elmt when i > j

if(word != ""){

↳ if(ans == ""){

ans = word;

word = ""

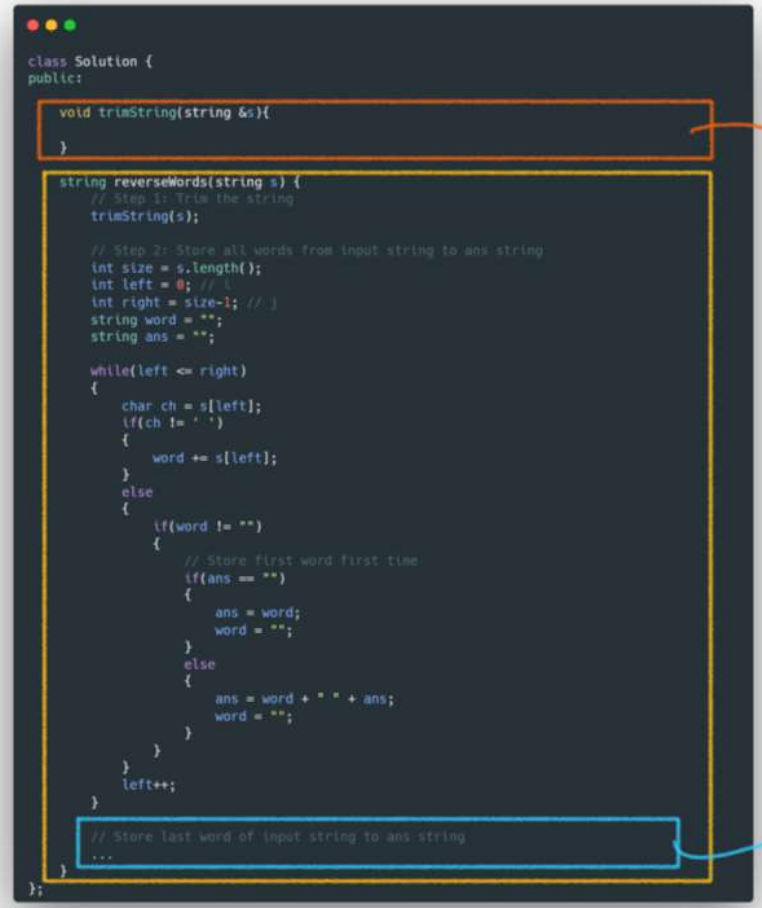
}

}

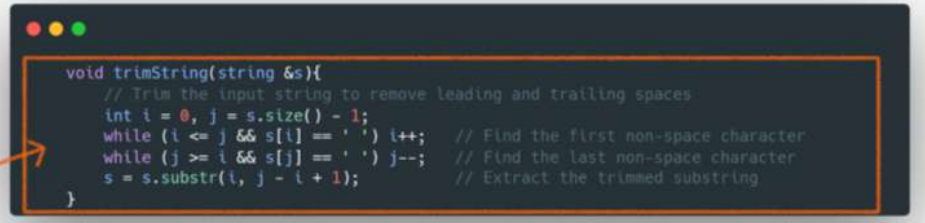
return ans

word = ""

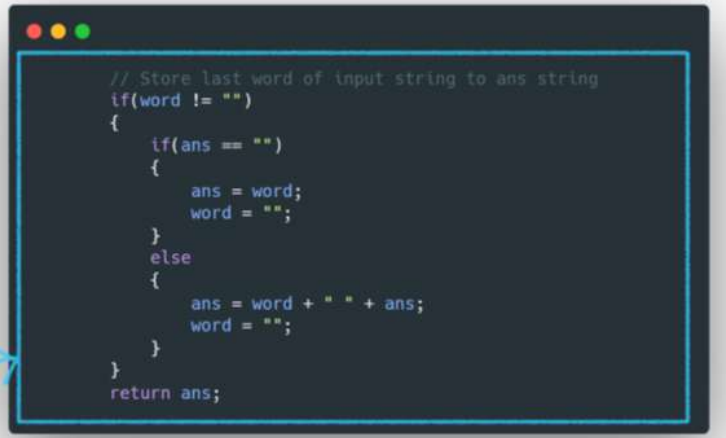
ans = "Good"



```
class Solution {
public:
    void trimString(string &s){

    }
    string reverseWords(string s) {
        // Step 1: Trim the string
        trimString(s);

        // Step 2: Store all words from input string to ans string
        int size = s.length();
        int left = 0; // i
        int right = size-1; // j
        string word = "";
        string ans = "";

        while(left <= right)
        {
            char ch = s[left];
            if(ch != ' ')
            {
                word += s[left];
            }
            else
            {
                if(word != "")
                {
                    // Store first word first time
                    if(ans == "")
                    {
                        ans = word;
                        word = "";
                    }
                    else
                    {
                        ans = word + " " + ans;
                        word = "";
                    }
                }
            }
            left++;
        }
        // Store last word of input string to ans string
        ...
    }
};
```



```
void trimString(string &s){
    // Trim the input string to remove leading and trailing spaces
    int i = 0, j = s.size() - 1;
    while (i <= j && s[i] == ' ') i++;   // Find the first non-space character
    while (j >= i && s[j] == ' ') j--;   // Find the last non-space character
    s = s.substr(i, j - i + 1);          // Extract the trimmed substring
}
```



```
// Store last word of input string to ans string
if(word != "")
{
    if(ans == "")
    {
        ans = word;
        word = "";
    }
    else
    {
        ans = word + " " + ans;
        word = "";
    }
}
return ans;
```