Problem 1 : Reverse Words in a String

🧠 **Problem Kya Tha?**
Given a string s, humein:

- Extra spaces ignore karte hue,
- Words ko split karna hai,
- Aur **poore string ke words ko reverse order me** print karna hai.

Example:
s = " hello world " → Output: "world hello"

Approach 1: using stringstream

📝 Code Breakdown + Explanation (Line-by-Line)

```
class Solution {
public:
    string reverseWords(string s) {
```

✔️ **Function reverseWords start** — input ek string s.

```
        stringstream ss(s);
```

✔️ **stringstream ss(s)**

- Yeh s ko ek stream ki tarah treat karta hai.
- Isse hum easily word by word string read kar sakte hai.

👉 **Why?**
Manual splitting me difficult hota hai extra spaces handle karna.
stringstream automatic ignore kar deta hai continuous spaces ko.

```
        string token ;
```

✔️ Ek temporary variable token.
Har extracted word yahan store hoga.
Example: Agar input = " hello world "
token values sequentially: "hello", "world"

```
        vector<string> words;
```

✔️ Words ko store karne ke liye ek vector banaya.
Baad me reverse karenge.

```
        while(getline(ss,token,' ')){
            if(!token.empty())
            words.push_back(token);
        }
```

✔️ **getline(ss, token, ' ')**

- Yeh stream ss se read karta hai

- ' ' space ko delimiter maan kar
- Har space ke beech ka part token me aa jata hai

👉 **Important:** Continuous spaces par token empty ho sakta hai.
Isliye:
`if(!token.empty())`

- Sirf non-empty word ko hi vector me daalte hai
- Taaki " " jaise extra spaces ignore ho jaye

Example:
Input: " a good example "
words vector banega:
["a", "good", "example"]

```
        int n = words.size();
```

✔️ Total words count le liya, reverse karne me kaam aayega.

```
        for(int i = 0; i <n/2; i++){
            swap(words[i] , words[n-i-1]);
        }
```

✔️ **Manual Reverse**

- First word ↔ last word
- Second ↔ second-last
- ... middle tak jaate hain

Example:
Before: ["hello", "world", "from", "cpp"]
After: ["cpp", "from", "world", "hello"]

```
        string result = "";
```

✔️ Final answer assemble karne ke liye empty string.

```
        for(int i = 0; i < n; i++){
            result += words[i];
            if(i != n-1) result += " ";
        }
```

✔️ Har word ko result me add kiya
✔️ Word ke baad **space add** kiya, but last word ke baad nahi.
Example output shape:
"world hello"
Jisme last me unnecessary space nahi hota.

```
        return result;
    }
};
```

✔️ Final reversed string return kar diya.

🎯 Final Output Example

Input:

```
"  hello   world  "
```

Processing Steps:

- words = ["hello", "world"]
- reverse = ["world", "hello"]
- result = "world hello"

⭐ Summary — Why This Approach Works?

- **stringstream** → easy splitting + handles extra spaces automatically
- **vector** → easy to store words
- **manual reverse** → no extra space usage for new vector
- **manual join** → no trailing space issue

Time Complexity: **O(n)**
Space Complexity: **O(n)** for storing words.

```cpp
class Solution {
public:
    string reverseWords(string s) {

        stringstream ss(s);
        string token ;

        vector<string> words;

        while(getline(ss,token,' ')){
            if(!token.empty())
            words.push_back(token);
        }

        int n = words.size();

        for(int i = 0; i <n/2; i++){
            swap(words[i] , words[n-i-1]);
        }
        string result = "";
        for(int i = 0; i < n; i++){
            result += words[i];
            if(i != n-1) result += " ";
        }
        return result;
    }
};
```

View less

Approach 2 : using istringstream

🔷 1. istringstream iss(s);

- Ye line **string s ko ek input stream (jaise cin)** ki tarah treat karti hai.
- Jaisa hum cin se ek-ek word read karte hain, iss se bhi waise hi kar sakte hain.

Example input:

```
"  hello   world from   cpp  "
```

🔷 2. iss >> token kya karta hai? — Sabse Important

>> operator (extraction operator) ka behaviour:

✔️ It automatically skips all leading spaces

Yaani agar string start me spaces hain → woh unko ignore kar deta hai.

✔️ It reads one word at a time

- Word = characters between spaces
- Jaise cin me hota hai: cin >> token

✔️ Continuous spaces → automatically skip ho jaate hain

Aapko empty token ka tension nahi.

🔥 Example Processing (iss >> token)

Input:

```
"  hello   world from   cpp  "
```

1st extraction

- Skip all spaces
- Read "hello"
  ➡️ token = "hello"

2nd extraction

- Skip " "
- Read "world"
  ➡️ token = "world"

3rd extraction

➡️ "from"

4th extraction

- Skip " "
  ➡️ "cpp"

No empty tokens!

vector becomes:

```
["hello", "world", "from", "cpp"]
```

🧠 Why this version is BETTER than getline?

✔️ getline(s, token, ' ')

- Continuous spaces → empty token produce karta hai → hume check lagana padta hai.

✔️ iss >> token

- Automatically spaces skip karega

- Sirf valid words hi extract karega
- No need for empty checks

Isliye ye version cleaner hai.

🔷 3. words.push_back(token);

- Jo bhi word token me aaya → vector me add kar diya.

🔷 4. Reverse Loop

```
for(int i = 0; i < n/2; i++){
    swap(words[i], words[n-i-1]);
}
```

- First ↔ last
- Second ↔ second-last
- Middle tak jaate hain

Before:
["hello", "world", "from", "cpp"]
After:
["cpp", "from", "world", "hello"]

🔷 5. Build final result string

```
result += words[i];
if(i != n-1) result += " ";
```

- Words ko dubara jod diya space ke sath
- Last word ke baad space nahi lagaya

Final output:

```
"cpp from world hello"
```

⭐ Full Summary (1 minute quick recall)

Concept	Meaning
istringstream iss(s)	string ko input stream bana deta hai
iss >> token	words ko ek-ek kar ke read karta hai
Space handling	All extra spaces automatically skip
No empty tokens	Iss >> token kabhi empty string nahi deta
Reverse logic	Simple swap approach
Result build	Words jod kar string banayi

```cpp
class Solution {
public:
    string reverseWords(string s) {

        istringstream iss(s);
        string token ;

        vector<string> words;

        while(iss >> token){

            words.push_back(token);
        }

        int n = words.size();

        for(int i = 0; i <n/2; i++){
            swap(words[i] , words[n-i-1]);
        }
        string result = "";
        for(int i = 0; i < n; i++){
            result += words[i];
            if(i != n-1) result += " ";
        }
        return result;
    }
};
```

Approach3:

✍️(Reverse Words — Smart Method)

✔ 1. int n = s.size();

- Bas input string ki length nikal li.
- Iska actual use iss code me nahi hai, par koi harm nahi.

✔ 2. stringstream ss(s);

Ye poori string ko ek stream me convert kar deta hai.

- Jaise hum cin >> token karte hain, waise hi ss >> token kar sakte hain.
- Automatic spaces skip ho jayenge.
- Har baar ek **word** milta hai.

Example:
Input:

```
"  hello   world from  cpp "
```

iss >> token se tokens:

```
"hello"
"world"
"from"
```

"cpp"
Empty tokens **kabhi nahi**.

✔ 3. string token = "";

- Temporary variable jisme har word aayega.

✔ 4. string result = "";

- Final answer yahan banega.
- Is approach ka main trick yahi hai.

⭐ MOST IMPORTANT PART

✔ 5. while (ss >> token)

❗ Meaning:

- Stream se **next word read karo**
- Jab tak words milte rahe loop chalega

🔥 Example:

Input:

```
"hello world from cpp"
```

Loop execution:

Iteration	token	result (after update)
1	"hello"	"hello "
2	"world"	"world hello "
3	"from"	"from world hello "
4	"cpp"	"cpp from world hello "

Notice:
Har new word **front me add** ho raha hai → Reverse automatically ho raha hai.

⭐ 6. Main Trick Line

```
result = token + " " + result;
```

✔ Why this reverses?

Kyuki har new word ko **starting me add** kar diya.

🔥 Example visual:

Start:

```
result = ""
```

1st token = "hello"

```
result = "hello "
```

2nd token = "world"

```
result = "world " + "hello "
       = "world hello "
```

3rd token = "from"

```
result = "from world hello "
```
4th token = "cpp"

```
result = "cpp from world hello "
```
Yeh bilkul stack jaisa behaviour hai — **Last read word first me aa raha**.

✔ 7. Remove last stray space

```
if(!result.empty() && result.back()==' '){
    result.pop_back();
}
```
- Last me ek space extra hoti hai
- Usko hata diya

Example:

```
"cpp from world hello "
                     ^ extra space
```
pop_back() → remove kar diya.

✔ 8. Return result

Final output mil gaya.

⭐ FINAL SUMMARY (30 sec recall)

- ss >> token = space skip + word extract
- Har word ko result ke starting me add kiya → auto reverse
- Last space remove ki
- Fast, easy, clean solution

Time: O(n)

Space: O(n)

```cpp
class Solution {
public:
    string reverseWords(string s) {
        int n = s.size();
        stringstream ss(s);
        string token="";
        string result="";

        while(ss >> token){//by default space ke baisis pr todega word k
            result = token +" "+result;
        }

        if(!result.empty() && result.back()==' '){
            result.pop_back();
        }
        return result;
    }
};
```

Approach 4 : interview friendly without stl

📌 **Reverse Words in a String (Without using built-in functions)**

🔷 **Problem:**
Given a string, reverse the order of words, remove extra spaces.
Example:

Input: " hello world meet me "
Output: "me meet world hello"

🔷 **Main Trick (Very Important 👇)**

Step 1: Reverse Entire String
Step 2: Reverse Each Word Individually
Example:

Original: "babua ds"
Reverse All: "sd aubaB"
Reverse word1: "ds"
Reverse word2: "babua"
Result: "ds babua"

🔷 **Why This Works?**
Instead of splitting → reversing list → joining,
hum **string ko ek plate ki tarah right se left palat dete hain**,
phir har word ko sahi direction me ghumate hain.

🧠 Step-By-Step Logic

Step	Task	Purpose
1	Trim leading + trailing spaces	Starting clean
2	Remove multiple spaces → single space	Normalized format
3	Reverse the whole string	Words ka order reverse ho jayega
4	Reverse each word in the reversed string	Final readable output

🧾 Handwritten-Style Pseudo Code (C++ Logic)

```
string s

1) trim: move left pointer until non-space
         move right pointer until non-space

2) normalize spaces:
   create new string result
   loop from left → right:
      if char != space → push
      else if last char in result != space → push space

3) reverse(result)

4) reverse each word:
   i = 0
   while i < result.length:
       j = i
       while j < n and result[j] != ' ':
             j++
       reverse substring from i → j-1
       i = j + 1
```

```cpp
class Solution {
public:
    void ReverseStr(string &s , int left , int right){
        while(left < right){
            char temp = s[left];
            s[left] = s[right];
            s[right] = temp;
            left++;
            right--;
        }
    }
    string reverseWords(string s) {
        int n = s.length();

        // 1.trim spaces from left and right
        int l  = 0 , r = n-1;
        while(l < n && s[l] == ' ')l++;
        while(r >= 0 && s[r] == ' ')r--;

        //2. Remove multiple spaces and build clean string
        string temp = "";
        while(l <= r){
            if(s[l] != ' ') temp.push_back(s[l]);
            else if(temp.back() != ' ') temp.push_back(' ');
            l++;
        }

        //3. Reverse the whole cleaned string
        ReverseStr(temp , 0 , temp.length()-1);

        //4. Reverse each individual word
        int start = 0;
        int len = temp.length();
        while(start < len){
            int end = start;
            while(end < len && temp[end] != ' ') end++;
            ReverseStr(temp,start,end-1);
            start = end+1;
        }

        return temp;
    }
};
```

⏱️ Time Complexity (TC)

Operation	Complexity

Trim	O(n)
Normalize spaces	O(n)
Reverse whole string	O(n)
Reverse each word	O(n)
Total	**O(n)**

📦 **Space Complexity (SC)**

Case	Space
If interviewer ignores temp string	O(1)
Otherwise (clean buffer used)	**O(n)**

🎯 **Summary (For Interview Answer)**
"I solve this by first cleaning spaces, then reversing the whole string and then reversing each word. This gives linear time complexity O(n) and constant extra space if we modify in-place. The trick is: **Reverse whole → then reverse individual words.**"