|  |  |
| --- | --- |
| **Valid Anagram in C++** | |
| #include <iostream>  #include <string>  #include <unordered\_map>  class ValidAnagrams {  public:  static bool sol(const std::string& s1, const std::string& s2) {  std::unordered\_map<char, int> map;  for (char ch : s1) {  map[ch]++;  }  for (char ch : s2) {  if (map.find(ch) == map.end()) {  return false;  } else if (map[ch] == 1) {  map.erase(ch);  } else {  map[ch]--;  }  }  return map.empty();  }  };  int main() {  std::string s1 = "abbcaad";  std::string s2 = "babacda";  std::cout << (ValidAnagrams::sol(s1, s2) ? "true" : "false") << std::endl;  return 0;  } | **Dry Run Table for ValidAnagrams::sol(s1, s2)**  **Input:**  s1 = "abbcaad";  s2 = "babacda";  **Step 1: Build Character Frequency Map (s1)**   | **Iteration** | **Character (ch)** | **map[ch] (Updated)** | **map State** | | --- | --- | --- | --- | | 0 | 'a' | 1 | { 'a': 1 } | | 1 | 'b' | 1 | { 'a': 1, 'b': 1 } | | 2 | 'b' | 2 | { 'a': 1, 'b': 2 } | | 3 | 'c' | 1 | { 'a': 1, 'b': 2, 'c': 1 } | | 4 | 'a' | 2 | { 'a': 2, 'b': 2, 'c': 1 } | | 5 | 'a' | 3 | { 'a': 3, 'b': 2, 'c': 1 } | | 6 | 'd' | 1 | { 'a': 3, 'b': 2, 'c': 1, 'd': 1 } |   Final **map** after processing s1:  { 'a': 3, 'b': 2, 'c': 1, 'd': 1 }  **Step 2: Validate Using s2**   | **Iteration** | **Character (ch)** | **Action** | **map[ch] (Updated)** | **map State** | | --- | --- | --- | --- | --- | | 0 | 'b' | Decrement | 1 | { 'a': 3, 'b': 1, 'c': 1, 'd': 1 } | | 1 | 'a' | Decrement | 2 | { 'a': 2, 'b': 1, 'c': 1, 'd': 1 } | | 2 | 'b' | Remove from map | — | { 'a': 2, 'c': 1, 'd': 1 } | | 3 | 'a' | Decrement | 1 | { 'a': 1, 'c': 1, 'd': 1 } | | 4 | 'c' | Remove from map | — | { 'a': 1, 'd': 1 } | | 5 | 'd' | Remove from map | — | { 'a': 1 } | | 6 | 'a' | Remove from map | — | {} |   Final **map** state: **Empty {}**, meaning both strings are anagrams.  ✅ **Output:** "true" |
| Output:- true | |