Check Palindrome in C++

```
#include <iostream>
#include <string>
using namespace std;

bool isStringPalindrome(const string& input, int s, int e) {
    // Base case: if start index equals end index, the string is a palindrome
    if (s == e) {
        return true;
    }
    // If the characters at the start and end do not match, it's not a palindrome
    if (input[s] != input[e]) {
        return false;
    }
    // If there are more characters to compare, call the function recursively
    if (s < e + 1) {
        return isStringPalindrome(input, s + 1, e - 1);
    }
    return true;
}

bool isStringPalindrome(const string& input) {
    int s = 0;
    int e = input.length() - 1;
    return isStringPalindrome(input, s, e);
}

int main() {
    cout << (isStringPalindrome("abba") ? "true" : "false") << endl;
    return 0;
}
```

Step-by-Step Function Call Flow:

1. **Initial Call:**
   isStringPalindrome("abba", 0, 3)
   - s = 0, e = 3
   - input[s] = 'a' and input[e] = 'a' → They match → Continue with the next call:

   isStringPalindrome("abba", 1, 2)

2. **Second Call:**
   isStringPalindrome("abba", 1, 2)
   - s = 1, e = 2
   - input[s] = 'b' and input[e] = 'b' → They match → Continue with the next call:

   isStringPalindrome("abba", 2, 1)

3. **Third Call (Base Case):**
   isStringPalindrome("abba", 2, 1)
   - s = 2, e = 1
   - Since s < e + 1 condition fails (2 > 1), the function returns true.
4. **Backtracking:**
   The result true propagates back through all the recursive calls:
   - isStringPalindrome("abba", 1, 2) → true
   - isStringPalindrome("abba", 0, 3) → true

Output:-
true