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;
}
```

Input

```
string = "abba"
```

🔍 Function Call Tree

```
isStringPalindrome("abba", 0, 3)
→ 'a' == 'a' ✓
→ isStringPalindrome("abba", 1, 2)
   → 'b' == 'b' ✓
   → isStringPalindrome("abba", 2, 1)
      → s > e → return true
```

📋 Dry Run Table

Call	**s**	**e**	**input[s]**	**input[e]**	**Match?**	**Return**
isStringPalindrome ("abba", 0, 3)	0	3	'a'	'a'	✓	✓
isStringPalindrome ("abba", 1, 2)	1	2	'b'	'b'	✓	✓
isStringPalindrome ("abba", 2, 1)	2	1	N/A	N/A	Base	✓

● Output

```
true
```

Your program will print:

```
true
```

Output:-

```
true
```