Longest Palindromic Substring

To solve the longest palindromic substring problem, the most efficient and intuitive approach is the Expand Around Center technique.

Key Idea: A palindrome mirrors around its center. So for each index in the string, try to expand outward to check if the substring is a palindrome.
There are 2n-1 centers to consider (because each character can be a center and the gaps between every two characters can also be a center for even-length palindromes).

Strategy: Expand Around Center:
1. Iterate through each character in the string.
2. For each index i:
   - Expand around center i (odd length palindrome)
   - Expand around center i and i+1 (even length palindrome)
3. Track the longest palindrome seen so far.

Helper Function: Write a function to expand from centers:

```
int left = i, right = i;
while (left >= 0 && right < s.length() && s[left] == s[right]){
    left--;
    right++;
}
```

Return the bounds of the longest substring found from that center.

Example: For s = "babad":
- Center at 1 (char 'a') → expands to "bab"
- Center between 1 and 2 → expands to "aba"

Return either "aba" or "bab"

Time Complexity:) · $O(n^2)$ time (n centers, each expands up to n)
· $O(1)$ space