problem statement

Longest Palindromic Substring Write a program to find the longest palindromic substring in a given string without using any built-in substring or reverse functions. For example, for input "babad", the output should be "bab" or "aba".

Code

```
function detectLargestSymmetricalChunk(text: string): string {
    let symmetryData = { position: 0, stretch: 0 };

    // Process each character and its surrounding neighbors
    for (let central = 0; central < text.length; central++) {
        // Examine odd-length symmetries (single character center)
        checkSymmetry(text, central, central, symmetryData);

        // Examine even-length symmetries (two-character center)
        checkSymmetry(text, central, central + 1, symmetryData);
    }

    return assembleSection(text, symmetryData.position, symmetryData.stretch);
}

function checkSymmetry(input: string, leftPos: number, rightPos: number, symmetryData: { position: number, stretch: number }): void {
    let leftTracker = leftPos;
    let rightTracker = rightPos;

    // Use a do-while loop for symmetry checking
    do {
        if (leftTracker < 0 || rightTracker >= input.length || input[leftTracker] !== input[rightTracker]) {
            break; // Exit the loop if symmetry breaks or boundaries are hit
        }
        leftTracker--;
        rightTracker++;
    } while (true); // Continue until break is triggered

    // Calculate the size of the current symmetrical chunk
    const currentSymmetry = rightTracker - leftTracker - 1;

    // Update the symmetry data if the current chunk is larger
    if (currentSymmetry > symmetryData.stretch) {
        symmetryData.position = leftTracker + 1;
        symmetryData.stretch = currentSymmetry;
    }
}

function assembleSection(content: string, startIndex: number, sectionLength: number): string {
    let chunk = "";
    for (let index = startIndex; index < startIndex + sectionLength; index++) {
        chunk += content[index];
    }

    return chunk;
}

const inputString = "babad";
console.log(detectLargestSymmetricalChunk(inputString));
```

Outputs: "bab"

Sample Test Cases

Testcase 1

"abaxyzzyxf" -> "xyzzyx"

Testcase 2

"abccbadad" -> "abccba"

Testcase 3

"xyzabcba" -> "abcba"