PSEUDOCODE FOR THE PYRAMID GENERATOR SCRIPT

Purpose:

Generate a symmetrical pyramid pattern using a specified character. The pyramid can have a defined number of rows, and the order of the rows can optionally be inverted.

---

1. Function: `padRow(rowNumber, rowCount)`

Purpose:

Generate a single row of the pyramid with the correct number of spaces and characters.

Input:

- `rowNumber`: The current row's number (1-based index).
- `rowCount`: The total number of rows in the pyramid.

Process:

1. Calculate the left padding (spaces before the character):
    - Use `" ".repeat(rowCount - rowNumber)` to add `(rowCount - rowNumber)` spaces.
2. Calculate the main content of the row:
    - Use `character.repeat(2 * rowNumber - 1)` to add `(2 * rowNumber - 1)` instances of the pyramid character.
3. Calculate the right padding (spaces after the character):

    - Use `" ".repeat(rowCount - rowNumber)` to add `(rowCount - rowNumber)` spaces.
4. Combine the left padding, main content, and right padding into a single string.

Output:

- Return the constructed row as a string.

---

2. Generate the Pyramid Rows

Purpose:

Use the `padRow` function to build all rows of the pyramid, either in normal or inverted order.

Input:

- `character`: The character used for the pyramid (e.g., `#`).
- `count`: The total number of rows in the pyramid.
- `inverted`: A boolean value that determines the order of rows:
    - `false`: Normal order (rows added to the end).
    - `true`: Inverted order (rows added to the beginning).

Process:

1. Initialize an empty array `rows` to store the rows of the pyramid.
2. Use a `for` loop to iterate from `1` to `count` (inclusive):
    - For each iteration:
        - If `inverted` is `true`:
            - Call `padRow(i, count)` and add the result to the beginning of the `rows` array using `rows.unshift()`.

        - Otherwise:
            - Call `padRow(i, count)` and add the result to the end of the `rows` array using `rows.push()`.

Output:

- The `rows` array containing all rows of the pyramid.

---

3. Construct the Result String

Purpose:

Format the rows into a complete pyramid string with a title.

Input:

- `rows`: An array containing all rows of the pyramid.

Process:

1. Initialize a string variable `result` with the title `"Ted's Pyramid Generator:"` followed by two newlines (`\n\n`).
2. Use a `for` loop to iterate over each row in the `rows` array:
    - For each iteration:
        - Append the current row and a newline character (`\n`) to the `result`.

Output:

- The formatted pyramid string.

---

4. Output the Result

Purpose:

Display the generated pyramid in the console.

Process:

1. Use `console.log(result)` to print the `result` string.

Main Execution Flow

Define Constants:

1. Set the constant `character` to `#` for the pyramid's symbol.
2. Set the constant `count` to `10` for the total number of rows.
3. Set the boolean `inverted` to `false` for normal row arrangement.

Generate Pyramid:

1. Call the `padRow` function for each row index (from `1` to `count`) using a `for` loop.
    - Append or prepend the result to the `rows` array based on the value of `inverted`.

Construct Final Output:

1. Build the `result` string with the formatted pyramid rows.

Display Output:

1. Use `console.log()` to print the pyramid to the console.

Optional Reverse Logic (Commented Out)

1. **While Loop**:
    - Continuously add rows to the `rows` array using `padRow` until its length equals `count`.
2. **For Loop (Reversed)**:
    - Iterate from `count` down to `1` to create the rows in reverse order.