

HASH AGILE

PROBLEM STATEMENT:

Write a program to generate and print the first n rows of Pascal's triangle without using built-in math or array functions. For n = 5, the output should be:

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

PROGRAM CODE:

```
function generatePascalsTriangle(n: number): void {
  let previousRow: number[] = []; // To store the previous row values

  for (let i = 0; i < n; i++) {
    let currentRow: number[] = new Array(i + 1).fill(1); // Initialize the current row with 1s

    // Compute intermediate values based on the previous row
    for (let j = 1; j < i; j++) {
      currentRow[j] = previousRow[j - 1] + previousRow[j];
    }

    // Convert row to a string with spaces between elements
    const rowString = currentRow.join(" ");

    // Calculate leading spaces to center the row
    const padding = " ".repeat(n - i - 1);

    // Print the formatted row
    console.log(padding + rowString);

    // Update the previous row for the next iteration
    previousRow = currentRow;
  }
}

// Call the function with n = 5
generatePascalsTriangle(5);
```

EXPLANATION OF THE CODE:

1. Function Definition

```
function generatePascalsTriangle(n: number): void {
```

This function generates and prints the first n rows of Pascal's Triangle.

2. Initialization of previousRow

```
let previousRow: number[] = [];
```

`previousRow` is used to store the values of the previous row in the triangle. It helps compute the current row iteratively.

3. Outer Loop (Rows)

```
for (let i = 0; i < n; i++) {
```

Iterates through each row from 0 to $n-1$. i represents the row index.

4. Creating the Current Row

```
let currentRow: number[] = new Array(i + 1).fill(1);
```

Initializes a new row with $i + 1$ elements, all set to 1.

The first and last elements of each row in Pascal's Triangle are always 1.

5. Filling Intermediate Values

```
for (let j = 1; j < i; j++) {  
    currentRow[j] = previousRow[j - 1] + previousRow[j];  
}
```

For rows beyond the first two, intermediate values are calculated based on the sum of two elements from the previous row.

6. Formatting the Output

```
const rowString = currentRow.join(" ");  
const padding = " ".repeat(n - i - 1);  
console.log(padding + rowString);  
currentRow.join(" ")
```

converts the row into a string with numbers separated by spaces.

padding adds spaces before the row to center-align it. The number of spaces decreases with each row. The row is printed with the padding added to the beginning.

7. Updating previousRow

```
previousRow = currentRow;
```

The currentRow becomes the previousRow for the next iteration.

COMPLEXITY ANALYSIS

Time Complexity:

The outer loop runs n times, and the inner loop computes each row's values with a total of $O(n^2)$ $2n \times (n-1) = O(n^2)$, including string formatting.

Space Complexity:

The algorithm uses $O(n)$ space for two arrays (current and previous rows) and padding.

OUTPUT

INPUT : $n = 5$

OUTPUT :

```
  1
 1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

The screenshot shows a web browser window with the URL `mycompiler.io/new/typescript`. The page title is "myCompiler". The language is set to "TypeScript". The code editor contains the following TypeScript code:

```
1 function generatePascalsTriangle(n: number): void {
2   let previousRow: number[] = []; // To store the previous row values
3
4   for (let i = 0; i < n; i++) {
5     let currentRow: number[] = new Array(i + 1).fill(1); // Initialize the current row with 1s
6
7     // Compute intermediate values based on the previous row
8     for (let j = 1; j < i; j++) {
9       currentRow[j] = previousRow[j - 1] + previousRow[j];
10    }
11
12    // Convert row to a string with spaces between elements
13    const rowString = currentRow.join(" ");
14
15    // Calculate leading spaces to center the row
16    const padding = " ".repeat(n - i - 1);
17
18    // Print the formatted row
19    console.log(padding + rowString);
20
21    // Update the previous row for the next iteration
22    previousRow = currentRow;
23  }
24 }
25
26 // Call the function with n = 5
27 generatePascalsTriangle(5);
```

The output of the program is displayed on the right side of the editor:

```
  1
 1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

The output is followed by the message: "[Execution complete with exit code 0]".

SAMPLE OUTPUT 1:

INPUT : n = 6

OUTPUT :

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
```

SCREENSHOT :

The screenshot shows the myCompiler.io website interface. The browser address bar displays "mycompiler.io/new/typescript". The page has a dark header with the "myCompiler" logo, language settings (English), and user options (Recent, Login, Sign up). Below the header, there is a text input field for a title and a dropdown menu set to "TypeScript". To the right of the code editor are "Run" and "Save" buttons. The code editor contains a TypeScript function named "generatePascalsTriangle" that takes a number "n" and returns a void. The function uses two arrays, "previousRow" and "currentRow", to calculate the values of the Pascal's Triangle. It iterates through each row, calculating the values based on the previous row's values. The output is displayed on the right side of the screen, showing the Pascal's Triangle for n=6. Below the output, it says "[Execution complete with exit code 0]". At the bottom of the page, there is a small advertisement for "Design and Development tips in your inbox. Every weekday."

```
1 function generatePascalsTriangle(n: number): void {
2   let previousRow: number[] = []; // To store the previous row values
3
4   for (let i = 0; i < n; i++) {
5     let currentRow: number[] = new Array(i + 1).fill(1); // Initialize the current row with 1s
6
7     // Compute intermediate values based on the previous row
8     for (let j = 1; j < i; j++) {
9       currentRow[j] = previousRow[j - 1] + previousRow[j];
10    }
11
12    // Convert row to a string with spaces between elements
13    const rowString = currentRow.join(" ");
14
15    // Calculate leading spaces to center the row
16    const padding = " ".repeat(n - i - 1);
17
18    // Print the formatted row
19    console.log(padding + rowString);
20
21    // Update the previous row for the next iteration
22    previousRow = currentRow;
23  }
24 }
25
26 // Call the function with n = 6
27 generatePascalsTriangle(6);
--
```

Program input

Output

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
```

[Execution complete with exit code 0]

Design and Development tips in your inbox. Every weekday.

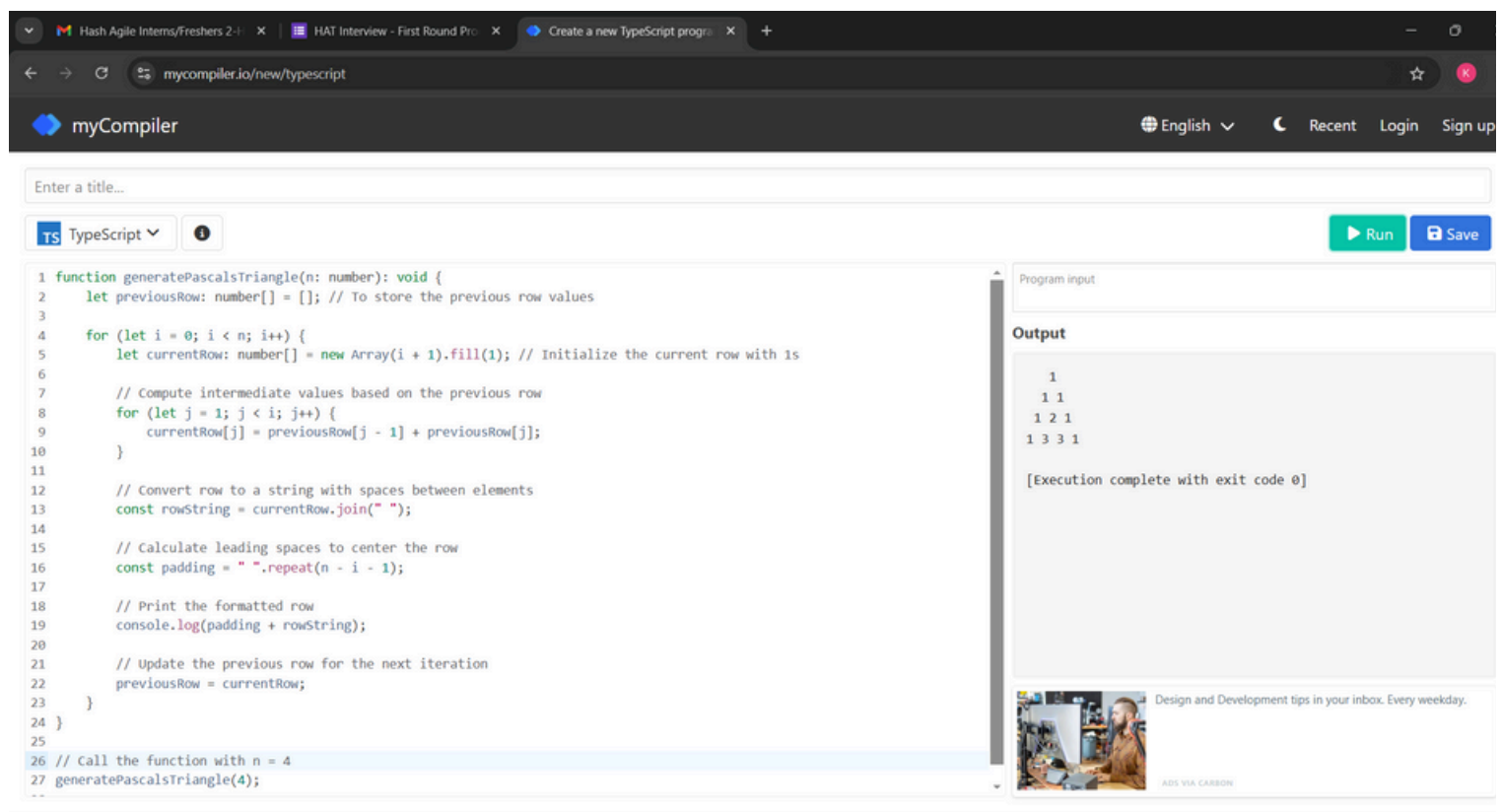
ADS VIA CARBON

SAMPLE OUTPUT 2 :

INPUT : n = 4

OUTPUT :
1
1 1
1 2 1
1 3 3 1

SCREENSHOT :



The screenshot shows the myCompiler.io web interface. The browser address bar displays "mycompiler.io/new/typescript". The page header includes the myCompiler logo, language settings (English), and user options (Recent, Login, Sign up). The main editor area is titled "Enter a title..." and contains a TypeScript file named "TS TypeScript". The code in the editor is as follows:

```
1 function generatePascalsTriangle(n: number): void {  
2   let previousRow: number[] = []; // To store the previous row values  
3  
4   for (let i = 0; i < n; i++) {  
5     let currentRow: number[] = new Array(i + 1).fill(1); // Initialize the current row with 1s  
6  
7     // Compute intermediate values based on the previous row  
8     for (let j = 1; j < i; j++) {  
9       currentRow[j] = previousRow[j - 1] + previousRow[j];  
10    }  
11  
12    // Convert row to a string with spaces between elements  
13    const rowString = currentRow.join(" ");  
14  
15    // Calculate leading spaces to center the row  
16    const padding = " ".repeat(n - i - 1);  
17  
18    // Print the formatted row  
19    console.log(padding + rowString);  
20  
21    // Update the previous row for the next iteration  
22    previousRow = currentRow;  
23  }  
24 }  
25  
26 // Call the function with n = 4  
27 generatePascalsTriangle(4);  
--
```

The output panel on the right shows the following output:

```
1  
1 1  
1 2 1  
1 3 3 1
```

Below the output, it states "[Execution complete with exit code 0]". At the bottom of the output panel, there is an advertisement for "ADS VIA CARBON" with the text "Design and Development tips in your inbox. Every weekday."

SAMPLE OUTPUT 3 :

INPUT : n = 7

OUTPUT :

```
      1
     11
    121
   1331
  14641
 15101051
1615201561
```

SCREENSHOT :

The screenshot shows a web browser window with the URL `mycompiler.io/new/typescript`. The page title is "myCompiler". The interface includes a "Run" button and a "Save" button. The code editor contains the following TypeScript code:

```
1 function generatePascalsTriangle(n: number): void {
2   let previousRow: number[] = []; // To store the previous row values
3
4   for (let i = 0; i < n; i++) {
5     let currentRow: number[] = new Array(i + 1).fill(1); // Initialize the current row with 1s
6
7     // Compute intermediate values based on the previous row
8     for (let j = 1; j < i; j++) {
9       currentRow[j] = previousRow[j - 1] + previousRow[j];
10    }
11
12    // Convert row to a string with spaces between elements
13    const rowString = currentRow.join(" ");
14
15    // Calculate leading spaces to center the row
16    const padding = " ".repeat(n - i - 1);
17
18    // Print the formatted row
19    console.log(padding + rowString);
20
21    // Update the previous row for the next iteration
22    previousRow = currentRow;
23  }
24 }
25
26 // Call the function with n = 7
27 generatePascalsTriangle(7);
--
```

The output panel shows the following Pascal's Triangle for n=7:

```
      1
     11
    121
   1331
  14641
 15101051
1615201561
```

Below the output, it says "[Execution complete with exit code 0]". At the bottom of the page, there is a footer with the text "Design and Development tips in your inbox. Every weekday." and a small image of a person.