**Recursive Implementaion**

// Algorithm: recursive\_candles

// Input: Array of candles, number of candles (n), current height, maximum height

MaxHeightCount recursive\_candles(candles, n, current\_height, max\_height)

{

// Base case: If there are no more candles

if (n == 0)

{

// Initialize result with height 0 and count 0

result ← {.height = 0, .count = 0};

return result;

}

// Initialize result with current height and count 0

result ← {.height = current\_height, .count = 0};

count ← 0;

// Count occurrences of the current height candle

for i ← 0 to n - 1 do

{

if candles[i] equals current\_height

{

count ← count + 1;

}

}

// Recursively call the function with rest of candles

rest\_of\_candles ← candles + 1;

recursive\_result ← recursive\_candles(rest\_of\_candles, n - 1, current\_height, max\_height);

// Update result count if current height count plus recursive count is greater

if count + recursive\_result.count > result.count

{

result.count ← count + recursive\_result.count;

}

// Update result height and count if recursive height is greater

if recursive\_result.height > result.height

{

result.height ← recursive\_result.height;

result.count ← recursive\_result.count;

}

return result;

}

// Algorithm: birthdayCandles

// Input: Array of candles, number of candles (n)

birthdayCandles(candles, n)

{

// Get the maximum height count recursively

max\_height\_count ← recursive\_candles(candles, n, candles[0], 0);

// Output the tallest candle height and count

write("The Tallest Candle Is: " + max\_height\_count.height);

write("Number of Tallest Candles: " + max\_height\_count.count);

return max\_height\_count.count;

}