**Recuesive Implementation 2**

// Algorithm: recursive\_birthdayCakeCandles2

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

MaxheightCount recursive\_birthdayCakeCandles2(candles, n)

{

// Initialize max\_height\_count with height 0 and count 0

MaxheightCount max\_height\_count = {.height = 0, .count = 0};

// Initialize stack to store indices of candles

int\* stack = allocate memory for an array of size n;

int top = -1;

// Loop through each candle

for (int i = 0; i < n; i++)

{

// Remove smaller candles from stack

while (top >= 0 && candles[stack[top]] < candles[i])

{

top--;

}

// If current candle is as tall as the top candle or stack is empty, push index to stack

if (top == -1 || candles[stack[top]] == candles[i])

{

top++;

stack[top] = i;

}

}

// Find the maximum height and count the number of tallest candles

int max\_height = candles[stack[0]];

int count = 1;

for (int i = 1; i <= top; i++)

{

if (candles[stack[i]] == max\_height)

{

count++;

}

else

{

break;

}

}

// Update max\_height\_count with maximum height and count

max\_height\_count.height = max\_height;

max\_height\_count.count = count;

// Free the memory allocated for stack

free(stack);

return max\_height\_count;

}