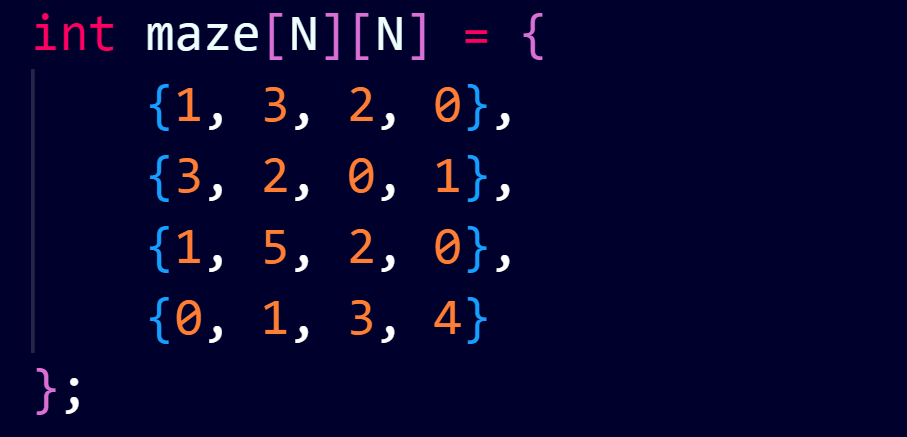
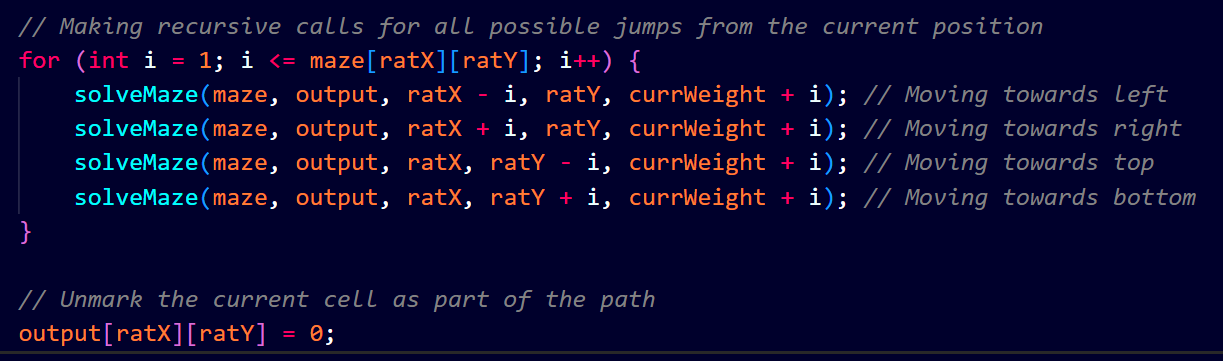
**MAZE**

**Declaration of 2D** **array**

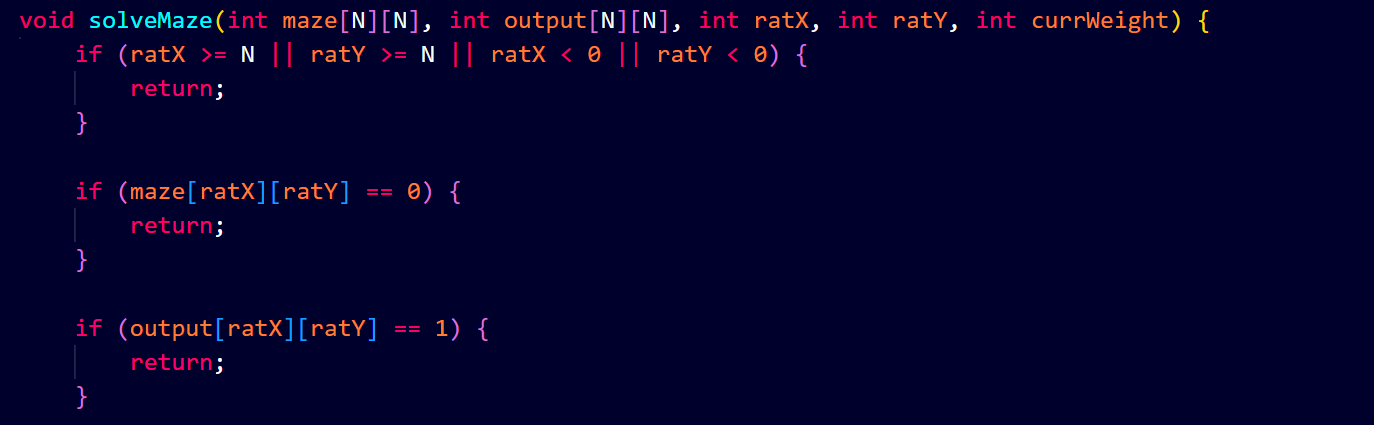


Here, the two-dimensional array (data-type: integer) represents the grids of the maze. The values 0 represent dead-end and non-zero numbers represent the maximum jumps that can be made from the specific cell.

**Backtracking to find longest path**

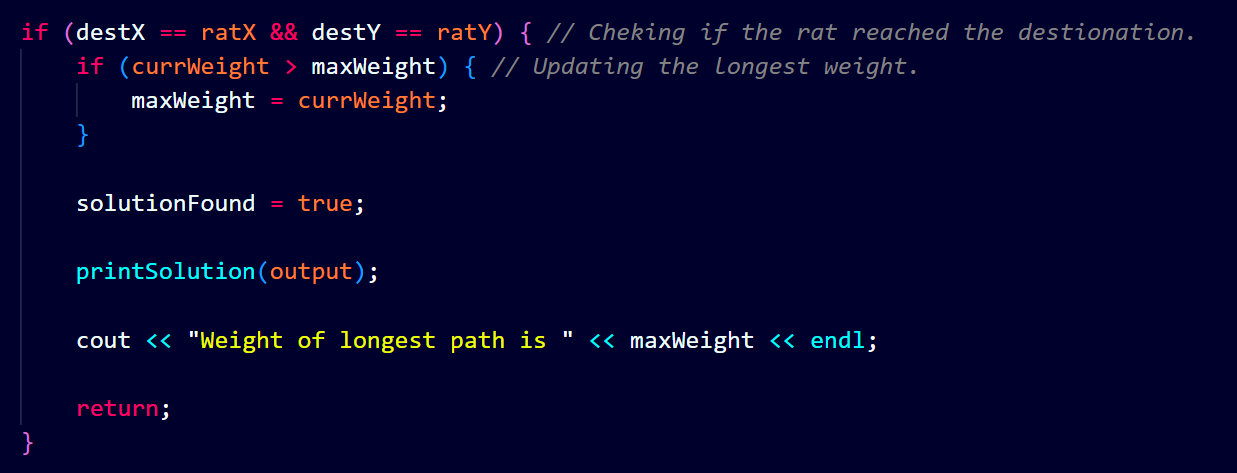


The solveMaze() function recursively travels towards all four directions.



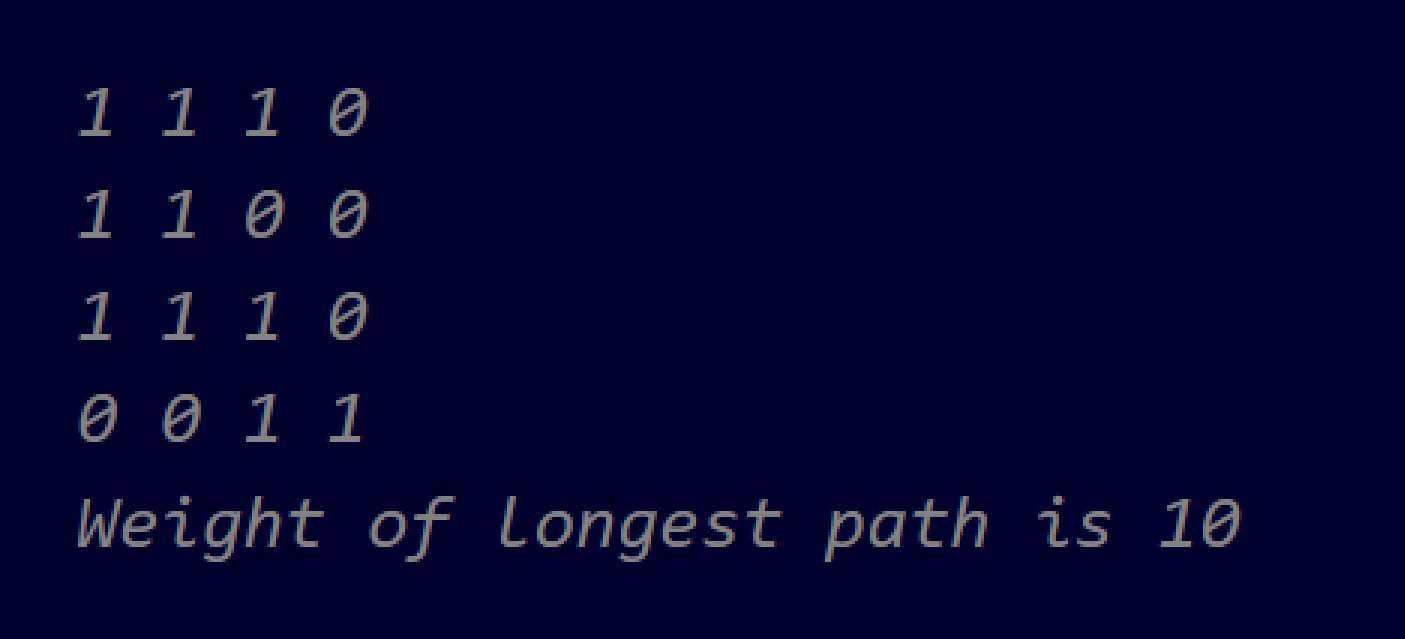
These are the conditions to stop travelling the out of matrix (maze), avoiding the visited cell and the paths that leads to dead-end. This makes a proper backtracking.

**Printing the solution matrix**



When solution is found (i.e., destination cell is reached), update the longest path and print solution matrix.

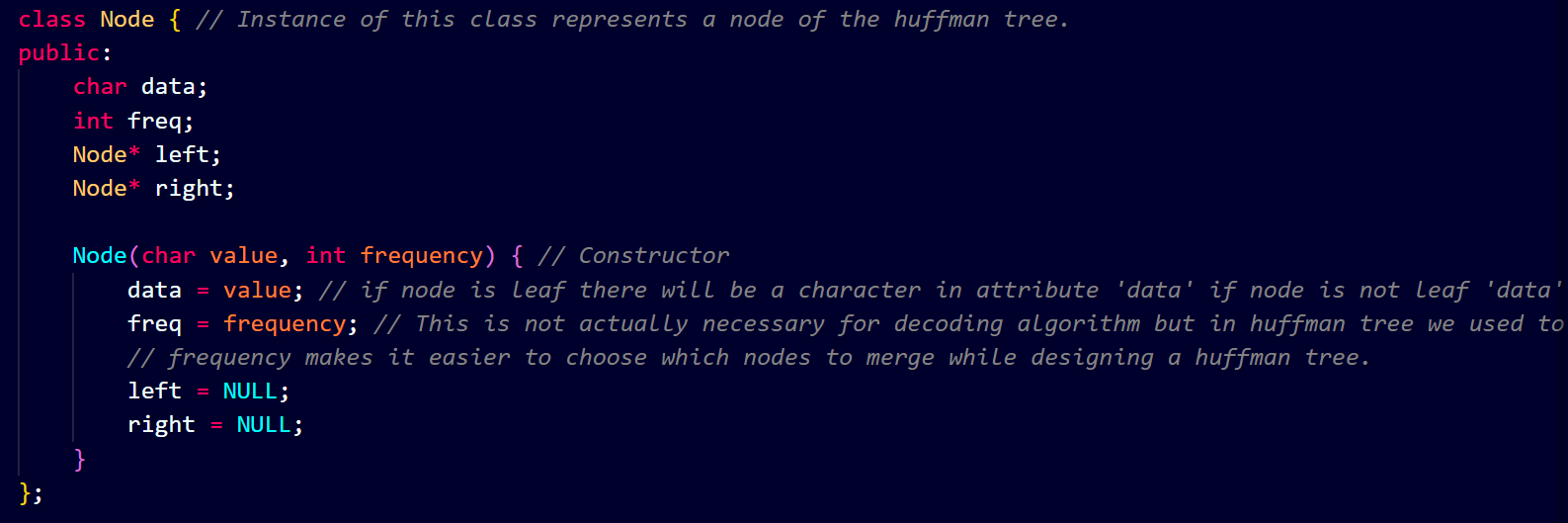
**Program output**



This is the output of program for the provided maze data.

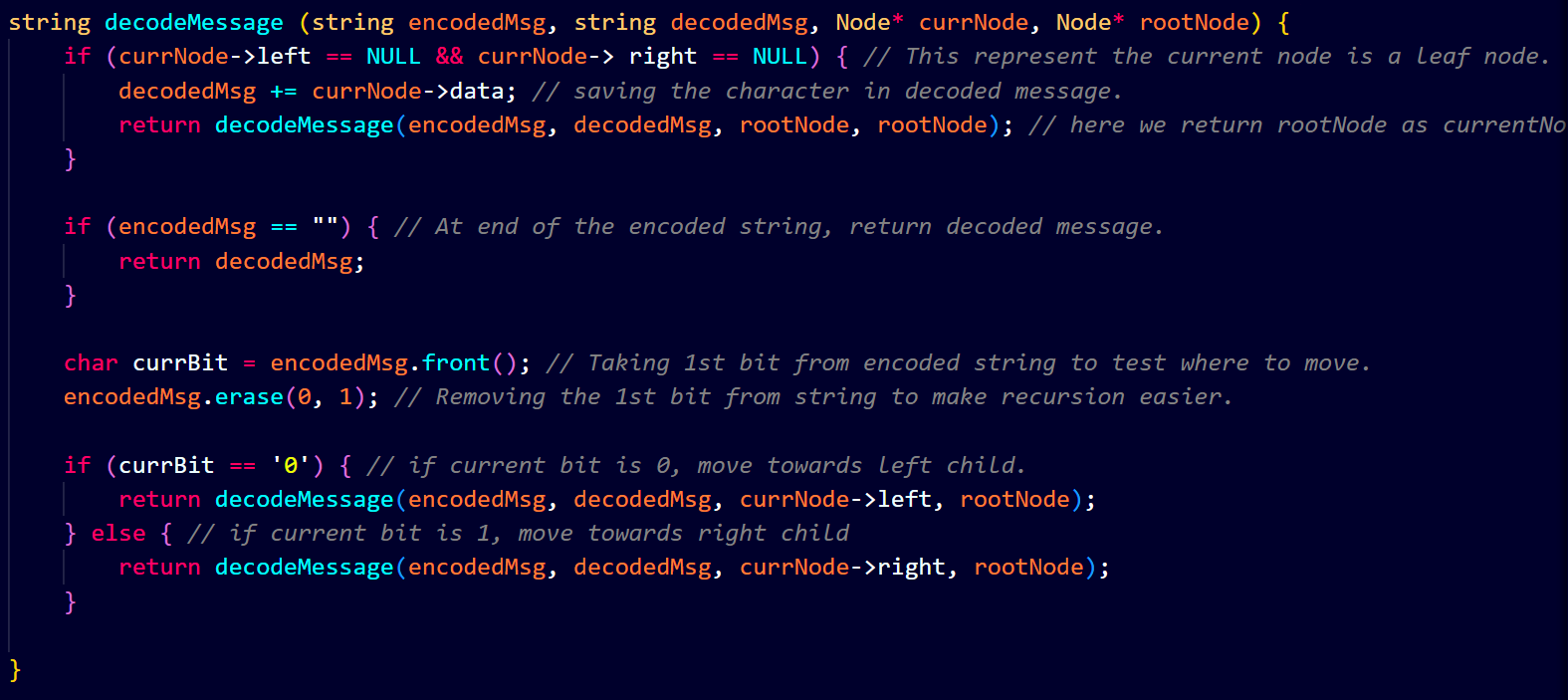
**Huffman Decoding**

**Declaration of node of Huffman tree**



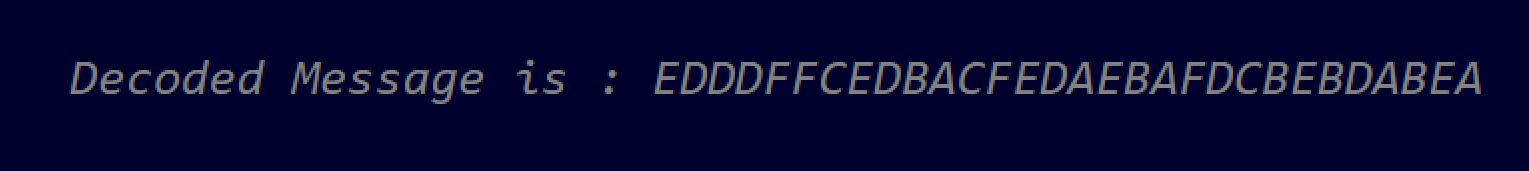
Here, the Huffman tree is represented as linked list. The node in this linked list has data, frequency, left and right pointer.

**Decoding the encoded message**



This is the decoder function that takes encoded message and root node of the Huffman tree and gives the decoded message. The basic concept is to move towards left or right child node of the Huffman tree for each bit in the encoded string until anyone of the leaf node is found.

**Output of program**



This is the output message for provided encoded string and Huffman tree.