**ICT208 Algorithms and Data Structures**

**Assessment 3: Recursion Case Study 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Linked Lists- Case study 1**  In this assessment, students will analyze a case and develop a solution by using recursive functions and finally write code. | Week 12 | 40% | 1,2,3,4 |

**Part A: Maze exploration using recursive function (30%)**

A Maze is given as N\*N binary matrix of blocks where source block is the upper left most block i.e., maze[0][0] and destination block is lower rightmost block i.e., maze[N 1][N-1]. A rat starts from the source and has to reach the destination. The rat can move in four directions: left, right, up or down.

In the maze matrix, 0 means the block is a dead end and 1 means the block can be used in the path from source to destination. Backtracking is an algorithmic technique for solving problems recursively by trying to build a solution incrementally in maze exploration.

**Suggested Approach:** Form a recursive function, which will follow a path and check if the path reaches the destination or not. If the path does not reach the destination then backtrack and try other paths.

**Algorithm:**

1. Create a solution matrix, initially filled with 0’s.

2. Create a recursive function, which takes initial matrix, output matrix and position of rat (i, j).

3. if the position is out of the matrix or the position is not valid then return.

4. Mark the position output[i][j] as 1 and check if the current position is destination or not. If destination is reached print the output matrix and return.

5. Recursively call for position (i+1, j) and (i, j+1).

6. Unmark position (i, j), i.e output[i][j] = 0.

**Your task** is to implement the given algorithm in C++ with recursive functions. You need to test the solution in a main function by creating a 2D array and print out the maze solution correspondingly if there is a path existing. Your main program should have an option that allows the program to find a solution using either 2 way movement or 4 way movement.

There are a few existing websites for reference on maze exploration using recursion for you:

https://www.geeksforgeeks.org/rat-in-a-maze-backtracking-2/

https://learn.saylor.org/mod/book/view.php?id=33001&chapterid=12855

https://runestone.academy/ns/books/published//pythonds/Recursion/ExploringaMaze .html

**Part B: Huffman decoding using recursive functions (10%)**

We have discussed Huffman encoding for data compression in lecture and tutorial, now we can implement the Huffman decoding for data extraction to recover the original data.

To decode the encoded data, we require the Huffman tree. We iterate through the binary encoded data. To find character corresponding to current bits, we use following simple steps.

1. We start from root and do following until a leaf is found.

2. If current bit is 0, we move to left node of the tree.

3. If the bit is 1, we move to right node of the tree.

4. If during traversal, we encounter a leaf node, we print character of that particular leaf node and then again continue the iteration of the encoded data starting from step 1.

**Your task** is to implement the Huffman decoding algorithm from the above steps in a C++ program with Huffman decoding function and a main function to decode a compressed string based on Huffman encoding and display the original string.

There are a few existing websites for reference on Huffman decoding algorithm for you:

https://www.geeksforgeeks.org/huffman-decoding/

https://www.codespeedy.com/huffman-decoding-in-cpp/

https://www.programiz.com/dsa/huffman-coding

Submission requirement: upload the Mazz.cpp and Huffman.cpp to moodle by 5pm, June 3, 2022.

**Marking rubric**

|  |  |  |
| --- | --- | --- |
| Marking criteria | Lecturer expectation | Marks |
| Declaration of 2D array in Mazz.cpp main function | 2D array contains correct data members with data types | 10 |

|  |  |  |
| --- | --- | --- |
| Backtracking of path in a maze in 4- direction | Backtracking  algorithms  implemented with recursive functions correctly comments | 45 |
| Display the exploration path in Maze | Maze path correctly implemented with comments | 20 |
| Decode the compressed Huffman code successfully in tree structure in Huffman.cpp | Huffman decoding is correctly  implemented with comments | 20 |
| Display the complete decoded string correctly in Huffman.cpp main function | Display is correctly implemented with comments | 5 |