**DSA ASSM 05**

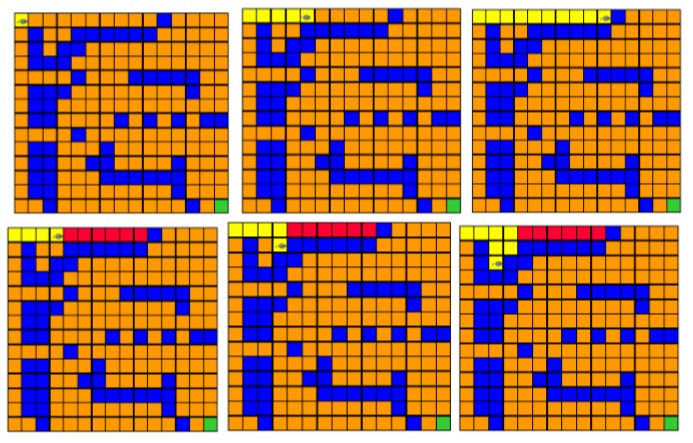
**Deadline: 21/06/2020**

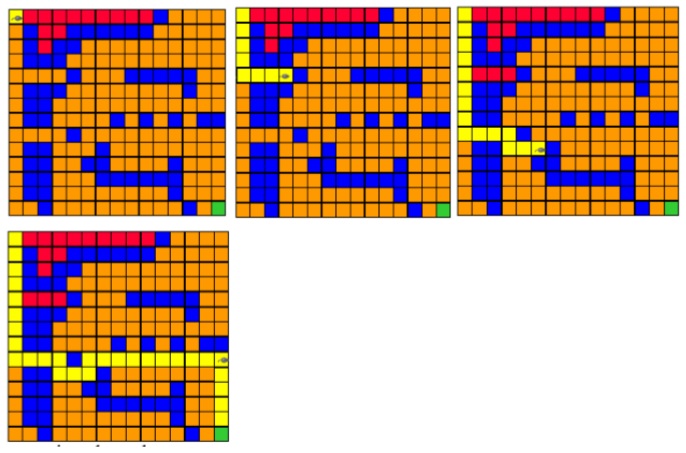
**Question 1:** Implement a template Stack class using Link List with one head pointer.

1. Push: Add an element in Stack
2. Pop: Removes top element from Stack
3. Retrieve: Returns value of the Top of stack element
4. Destructor
5. IsFull
6. IsEmpty

**Rat in a Maze Problem.**

Write a program that uses a stack to solve the maze problem. It will use 2-D dynamic array (as maze), which dimensions will be taken as input from user. Following figures show layout of maze. You do not need to implement the graphical user interface but you must have to output 2-D board on command line.





Here are some guide lines for games.

1. First step for the rat is to enter in maze (you can set that to First Square [0][0]).
2. Orange and green squares in figure are squares in which the rat can move.
3. Blue squares are hurdles and rat cannot move there. You can use any marking colors or values (0, 1, 2) for your game.
4. Move order is just in four directions: Right, Down, Left, Up
5. Block visited squares to avoid revisit them by any marker value (2) or color (as shown red in figure).
6. Path from maze entry to any current position of rat must operates as a stack.
7. At the end the path from cheese (you can fix that as last square) till entry cell should be marked and printed in 2-D maze matrix.
8. Placement of Hurdles should not be fix it should change with each instance of new game.

**Question 2:** Implementation of Huffman Encoding using Heaps and Binary Trees

Implement a class Huffman which contains the root Node of Huffman tree. Each node in Huffman is of type HNode.

struct HNode

{

int freq;

char character;

HNode \*left, \*right;

}

**Note:**

1. A valid character is stored only in leaf nodes in a Huffman Tree.
2. In a leaf node, the value in freq variable corresponds to the frequency of the character that the node represents.
3. In a non-leaf node, the value in freq variable is the sum of freq variables stored in its left child and right child.
4. Implement a member function createHuffman that is passed as parameter a filename (file is uploaded on portal too). The file contains characters along with their frequencies. The function creates a Huffman tree of the given characters. You will need MinHeap data structure for this. Use your previous MinHeap implementation. void createHuffman(string const filename)
5. Implement a member function printHuffman which prints the Huffman code of each character along with the character and its frequency. printHuffman() const
6. Destructor
7. Create a main function which creates a Huffman object, calls the createHuffman function, and then calls the printHuffman function to print Huffman code.