## Instructions:

## Computing 3: 2018/2019

## Joint DSA & Programming Assignment

## Deadline 26/11/18

## 1. Write an application to implement “2D Maze” as described below.

## 2. The solution is to be presented in a report. This report is to contain

## a thorough description of the problem,

## a description of underlying data structures used to solve the problem,

## pseudocode of the Algorithms used,

## a copy of the C++ code,

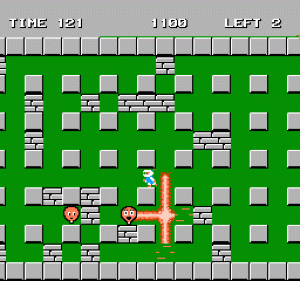
## a description of all the functions/routines which have been used, and

## Test data used and sample execution screenshots of outputs produced.

## 3. A copy of the code and an executable is to be placed on your M: drive in a folder called DS-Ass1. A copy of the code must also be uploaded blackboard. You will be required to demo your application for assessment after the due date.

## 4. Due Date: Monday 26th November @5pm.

2 D Maze



Sample Screen ( not actual game)

**Components:**

# 1: The maze

Store the 2d maze as a file. The file must store the size of the maze in rows and columns. It must store whether each array element is either a corridor or a wall. The file must be able to store 1 maze of at least 10 by 20, and store the number of enemies per maze.

The maze is stored as an array of ints. The file must be store 1 maze. Look at Dynamic 2D array in Labs 2018 in blackboard.

Load in the file and draw the maze.

# 2: The Player

The player is represented by an instance of a Player class (you create) that either inherits from sf::Sprite or has a sf::Sprite object as a member variable.

The player character is drawn as a sprite that moves left, right, up and down along the corridors. The player cannot walk through walls. The player moves from one grid position into the adjacent one.

The player moves from one grid position to the next and must not move too fast.

**Note**: You must use sf::Text to display the player’s lives and scores.

# 3: The Enemies

Each enemy is represented by an instance of an Enemy class (you create) that either inherits from sf::Sprite or has a sf::Sprite object as a member variable.

The enemies are stored as an array of Enemy objects.

At the start of each game the enemies are randomly placed in the corridors (anything that’s not a wall). The enemies cannot be placed on a grid position that already contains a player or another enemy.

The enemy randomly chooses a direction to move. If there isn't a wall in its way it will move into to the next grid position, the enemies must not move too fast.

When an enemy dies it is brought back to life and is repositioned in a random corridor, not on top of the player or another enemy.

# 4: The Bomb (or a similar non-violent equivalent)

The bomb (only 1 bomb) is represented by an instance of a bomb class (you create).

The player can drop a bomb; the bomb has a fuse of 3 seconds. During this time the player cannot change the bombs position. When the bomb explodes it, it explodes in 8 directions, the diagonals and left, right, up and down. This explosion is instantaneous, it destroys any enemy or player within a distance of 4 grid tiles in those directions. The bomb however cannot go through the walls i.e. a player or enemy with a wall between it and the bomb will be unaffected.  The player’s score is increased for each enemy destroyed.

# 5. Winning and Losing

A player wins when a set number of enemies (e.g. 10) have been destroyed. The game is lost when the players lose all their lives.

# Note: Collision Detection

If the player collides (intersects) with an enemy, the player loses a life. If the player loses all lives, a “Game over” sprite is shown.