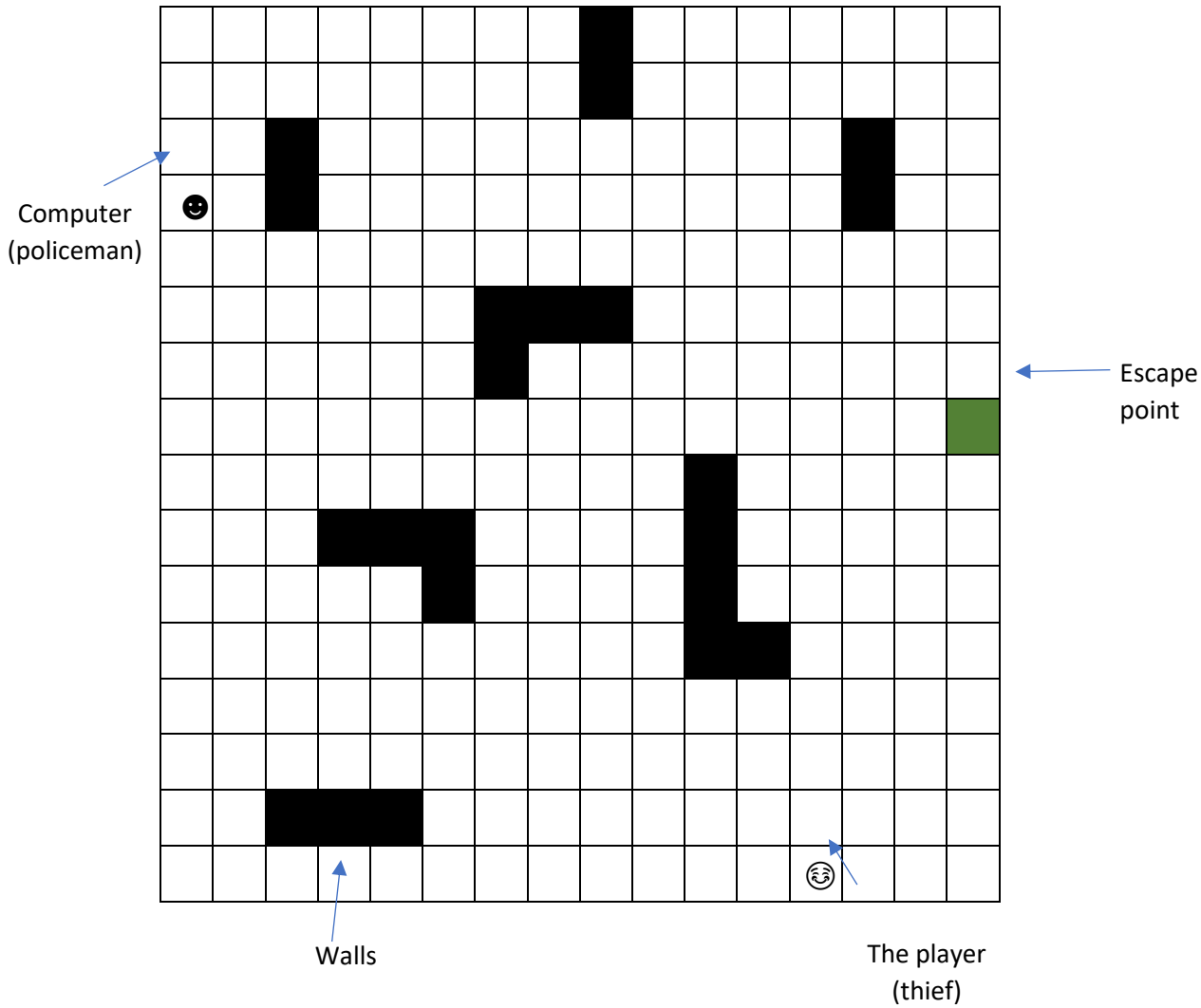
# EHB110E 2019-2020 Fall Project

In this project, you will be programming a game played against a computer. The game will be played on a 16x16 grid maze as shown below. You have to place the walls within the maze as shown below.

You will be a thief trying to escape from the maze at the escape point before being caught by a policeman (the computer) chasing you.




The player and policeman are allowed to move up, down, left and right. No diagonal moves are allowed. The player and the policeman cannot go through the walls and must move around them.

The game is played in turns. The player makes his/her move and then the computer makes its move.

The escape point will be on the 8th row and 16th column of the maze as shown in the figure. If the player reaches this square, the player wins the game. If, otherwise, the policeman reaches the square where the player currently resides, the game is lost.

At the beginning of the game, the player and the policeman are randomly placed on the maze not closer than **16 squares** from each other and the thief not closer than **16 squares** from the escape point in **Manhattan distance**.

The Manhattan distance between two squares at coordinates (x1, y1) and (x2, y2) is defined as |x2-x1| + |y2-y1|

The game will have two difficulty levels: easy and hard. In the easy level, the policeman will make a random move to one of the feasible squares at each turn. A feasible square is a square that is adjacent to the current square the policeman resides and that doesn't have a wall in it.

In the hard level, the computer will be smarter and will make a move to an **optimal** feasible square. An optimal feasible square is the square that makes the distance to the thief the shortest. If two or more squares have an equal distance, the square to move to will be picked randomly among the optimal feasible squares. You may use any algorithm you desire to find the optimal feasible squares. Hint: Don't let the policeman fall into an infinite loop by checking the last position the policeman was at before.

The game will first ask for the difficulty level. After the difficulty level is selected, the game will start with the player's move.

When the current game ends, a message that tells "You lost!" or "You won!" will be displayed on the screen depending on the outcome. Then, the player will be asked if she/he wants to play the game once more or wants to exit the game. If the player chooses to play once more, the above actions will repeat starting from choosing the difficulty level again. If the player decides to exit, the program will end.

## Project Deliverables

1. You should submit your source code. Your source code should be commented.

2. You should include a text file named README that explains how to compile your source code and how to run it. Your README file should include an author section that has your name in it. It should also have a version section which lists the current version of your program, such as version 1.0.

3. Although not needed, you may include a binary, but your code will be tested whether it compiles and runs correctly.

## Tips for the Project

1. You may use ASCII characters _, |, W, P, T to represent maze square boundaries, walls, the policeman and the thief, respectively when drawing the game platform to screen.

2. You can represent the maze by a 16x16 two-dimensional array. Each position in the array may represent either an empty square, a wall, the policeman or the thief.

3. You can use system("clear") command to clear the screen when refreshing the maze