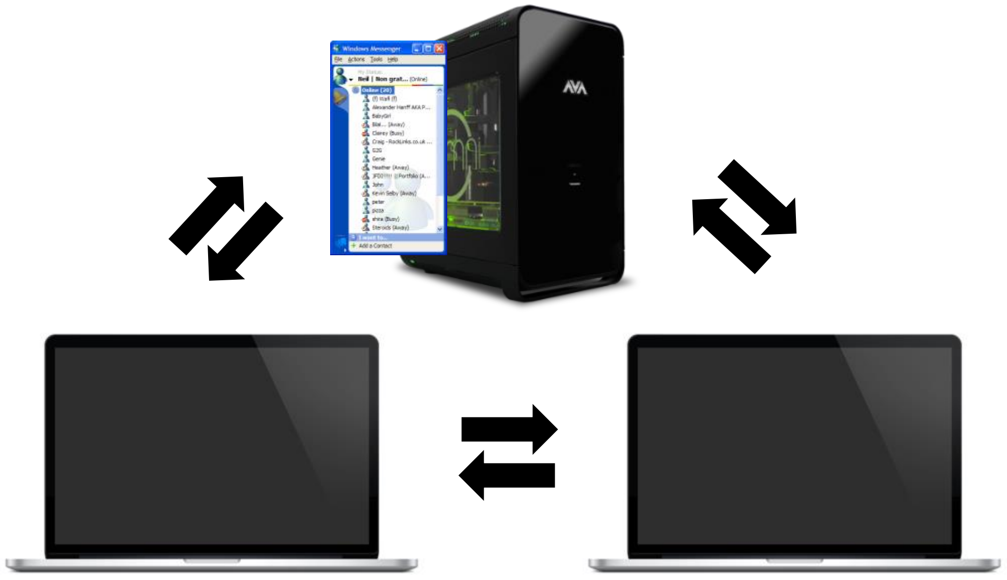
Escape Plan

An aim of this assignment is to show how to implement a simple socket programing client-server application. In order to develop ***Escape Plan*** game, it has to be implemented based on socket programming. The game client must be deployed on two computers. Users can play the game between two or more connected computers as a regular online application.

a. **Client-Server:** Students need to implement a server that waits for client connection. Then when a client is on the network, it would connect to the server first and get information about another client. Therefore, the client knows who is on the same network at that time. In this case, one of computers has a server program and also a game client. Another computer has only the game client that will directly connect to the server.

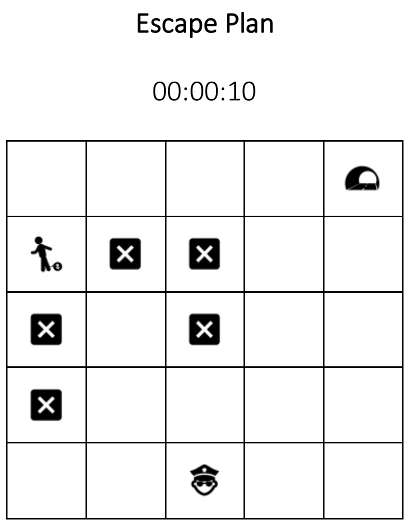


Requirement:

- Assume each client game knows what server's address and also server's port are. Game clients **do not require** IP address and port number filling. Server's IP and port should be set in your program's source code.
- The server program shows the number of concurrent clients and clients that are online.
- In the 5x5 blocks of the map in this game, there are 3 types of blocks including 19 ***free blocks***, 5 ***obstacle blocks*** (or 20% of map size) and 1 ***tunnel block***. Moreover, there are two characters in the game including a ***warder*** and a ***prisoner***.
- Every ***free block*** must be accessible. The ***warder*** and the ***prisoner*** cannot access the obstacle blocks and only the ***prisoner*** can access the ***tunnel block***.
- When the game starts, the server places all the blocks randomly and then places the characters randomly in the free blocks.

- The server picks a character randomly for a player. The player who becomes the ***warder*** will start to move first.
- For each turn, the players have only 10 seconds to move their characters to one of the adjacent blocks
- If the ***warder*** accesses to the same block of the prisoner, the player who plays the ***warder*** will win. On the other hand, if the ***prisoner*** accesses the ***tunnel*** block, the player who plays the ***prisoner*** will win.
- The game then pops up the winner's name and the current scores of both of the players. The winner will be the first player for the next game.
- The server has a reset button to reset the game and the players' scores.

Example User Interface:



Hint: You can implement in any programming languages and make your own creative design.
http://www.tutorialspoint.com/javaexamples/net_multisoc.htm
http://java.sun.com/developer/onlineTraining/Programming/BasicJava2/socket.html
http://www.javaworld.com/article/2077322/core-java/sockets-programming-in-java-a-tutorial.html
https://nodejs.dev/learn
https://docs.aws.amazon.com/apigateway/index.html
https://www.vpn.net/

Score Criteria (Full score = 25 points)

a) (5.0) Demo preparation and creativity
b) (10.0) Fundamental implementation
c) (10.0) Extra features

Escape Plan (Fundamental implementation)

Implement server and game client.

(0.5) One of computer has server program and also game client.
(0.5) Another computer has only game client that will directly connect to server.

Client

(1.0) Client connects to server first and gets information about other clients from server program.
(1.0) Each client knows what server's address and server's port are. Server's IP and port will be set in your program's source code.
(0.5) Can put your nickname when the game starts.
(0.5) Welcome message appears on the game starts.
(0.5) Every free block can be accessible.
(0.5) Obstacle block cannot be accessible.
(0.5) The warder cannot access to the tunnel block.
(0.5) Time can be counted down.
(0.5) Character can move to only adjacent block.
(0.5) The current game end when the warder or prisoner reach their objective.

Server

(1.0) Server program shows the number of concurrent clients that are online.
(1.0) Server has a reset button to reset player's scores and current game.
(1.0) Server randomizes map and character for player.

Extra features (1.0 per feature)

1	6
2	7
3	8
4	9
5	10