The Class Board

1 Specification:

1) The game is played on a rectangular board consisting of X by Y squares - both values are freely selectable by the player at the beginning of each game.
2) Make sure your board (and their implementation) can handle any size.
3) Each square can hold an item, an enemy or be empty.
4) At the beginning of a game the board is initialized, i.e. each square is assigned an enemy, an item or stays empty.(we use the equal chance to assign them).
5) The game is based on simple commands entered via the keyboard.
6) After choosing a character the player begins the game at the starting square.
7) Using look function to print out information about the current location.

1.1 Specification of Board Generation

1. **Board ():** The default constructor.
   Pre: none.
   Post: create new empty board. x=y=0 and allocate resource. If there is no resource it will emerge the exception.

2. **Board (NewX, NewY):** create a dynamic game board size.
   Pre: the "x" and "y" will get form the user.
   Post: create a fix board with the X and Y. If there is no resource it will emerge the exception.

3. **~board ()**: destructor.
   Pre: none.
   Post: delete the game board and release resource.

4. **board (orig):** copy constructor to create a dynamic array.
   Pre: *There has been a board class.*
   Post: copy another board class and allocate *resource. Throws an exception on failure*

5. **board &operator=(orig):**assignment operate
   Pre: There has been a board class.
   Post: overload the assignment operate and allocate resource when use "=". Throws an exception on failure

6. **BoardInitialization ()**: initialized the board use the private argument x and y.
   Pre: there has created the game board with the x and y.

Post: each square can hold an item, an enemy or be empty; each square is assigned an enemy, an item or stays empty

7. **SetPlace(x, y, items):** set the items to the appointed place
   Pre: There has created the game board with the x and y.
   Post: set the items to the square [x][y].

8. **NextPlace(x, y):** move to the appointed square.
   Pre: have created the board.
   Post: use the argument x and y to move to the square [x][y].

9. **BoardLook(x, y)**: get the number of the current square.
   Pre: have created the board.
   Post: use the argument x and y to get the value of the square [x][y].

2 User's View of Services

- board();
- board(unsigned int NewX, unsigned int NewY);
- virtual ~board();
- board (board &orig);
- board &operator=(const board &orig);
- void BoardInitialization();
- void SetPlace(unsigned int &x, unsigned int &y, unsigned int &i);
- inline void NextPlace(int &NextX, int &NextY);
- inline unsigned int BoardLook();

3 Internal Data Representation

Variable	Type	Access
x	Unsigned Int	private
y	Unsigned Int	private
**CurrentState	Unsigned Int	private

4 Remaining Definitions

5 Coding