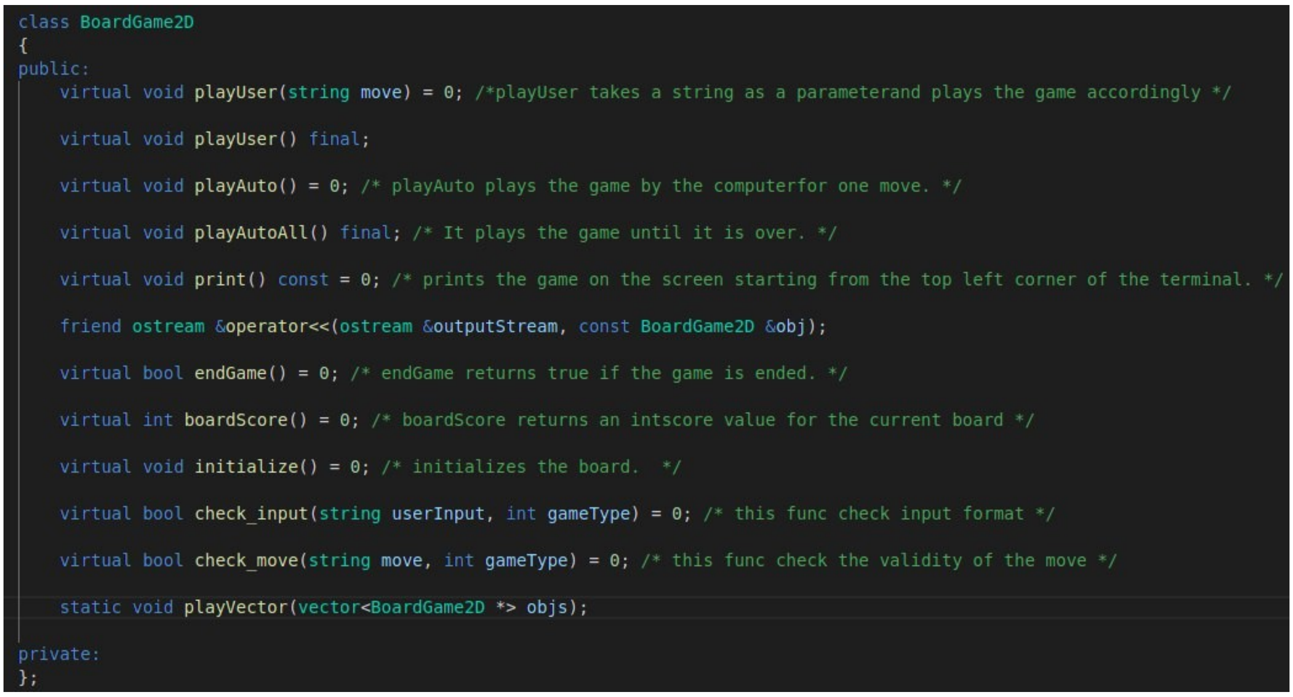
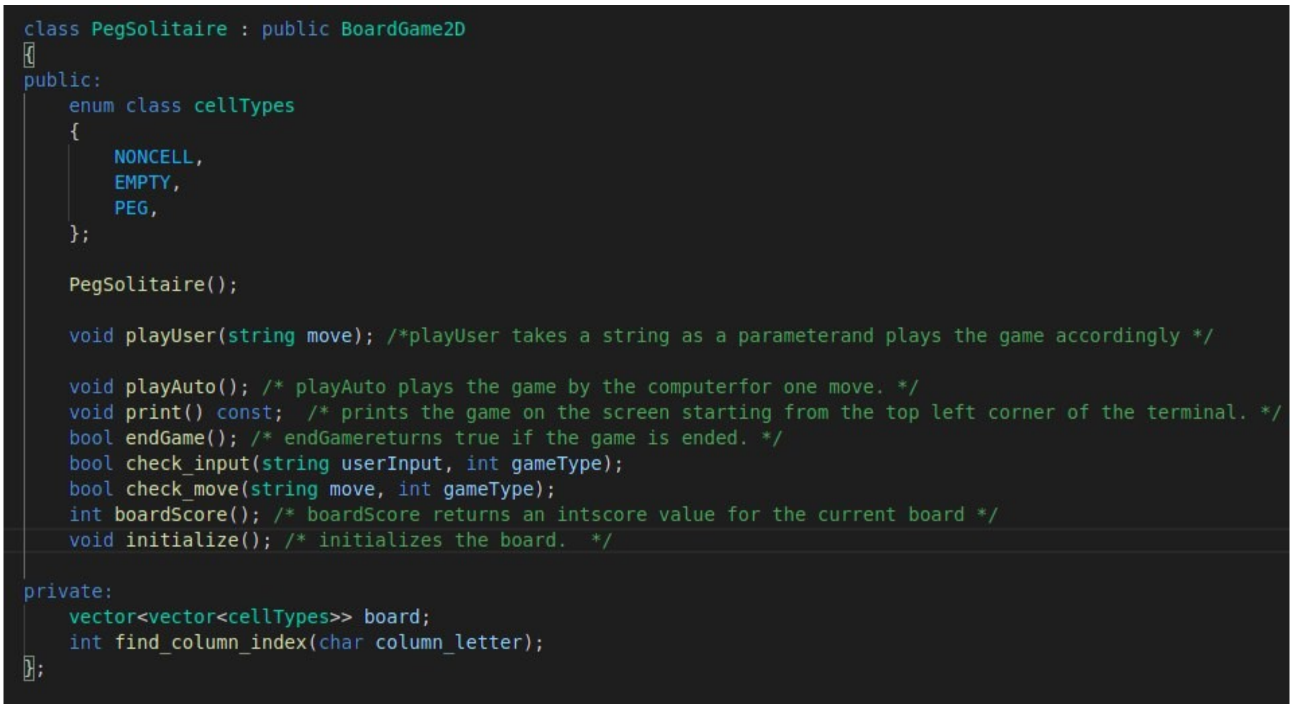

```
class BoardGame2D
{
public:
    virtual void playUser(string move) = 0; /*playUser takes a string as a parameterand plays the game accordingly */

    virtual void playUser() final;

    virtual void playAuto() = 0; /* playAuto plays the game by the computerfor one move. */

    virtual void playAutoAll() final; /* It plays the game until it is over. */

    virtual void print() const = 0; /* prints the game on the screen starting from the top left corner of the terminal. */

    friend ostream &operator<<(ostream &outputStream, const BoardGame2D &obj);

    virtual bool endGame() = 0; /* endGame returns true if the game is ended. */

    virtual int boardScore() = 0; /* boardScore returns an intscore value for the current board */

    virtual void initialize() = 0; /* initializes the board.  */

    virtual bool check_input(string userInput, int gameType) = 0; /* this func check input format */

    virtual bool check_move(string move, int gameType) = 0; /* this func check the validity of the move */

    static void playVector(vector<BoardGame2D *> objs);

private:
};
```

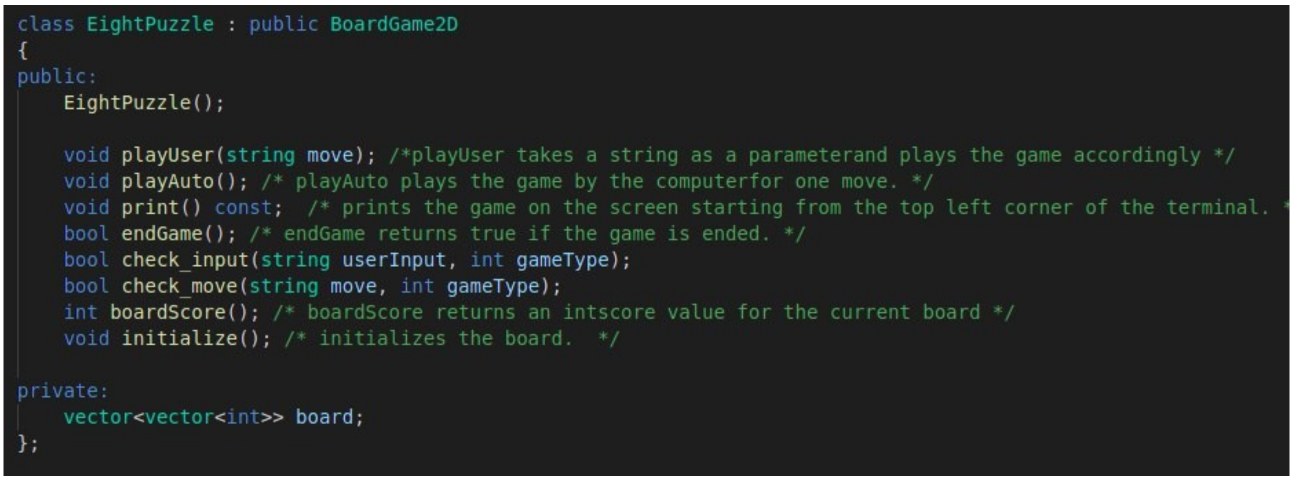
BoardGame2D class interface.
I added 2 more pure virtual functions.
Check input , checks the input format.
Check move, checks the validity of the move.


```
class PegSolitaire : public BoardGame2D
{
public:
    enum class cellTypes
    {
        NONCELL,
        EMPTY,
        PEG,
    };

    PegSolitaire();

    void playUser(string move); /*playUser takes a string as a parameterand plays the game accordingly */

    void playAuto(); /* playAuto plays the game by the computerfor one move. */
    void print() const;  /* prints the game on the screen starting from the top left corner of the terminal. */
    bool endGame(); /* endGamereturns true if the game is ended. */
    bool check_input(string userInput, int gameType);
    bool check_move(string move, int gameType);
    int boardScore(); /* boardScore returns an intscore value for the current board */
    void initialize(); /* initializes the board.  */

private:
    vector<vector<cellTypes>> board;
    int find_column_index(char column_letter);
};
```

I have a enum class, which represent the every cell of PegSolitaire.
VALID MOVE TYPES FOR PEG SOLITAIRE ARE:
A3-D , B2-u, c3-d

WHAT I RETURN FROM BOARDSCORE IN PEG SOLITAIRE IS:
The number of pegs that left on the board.
Basically, it counts remain peg and returns it

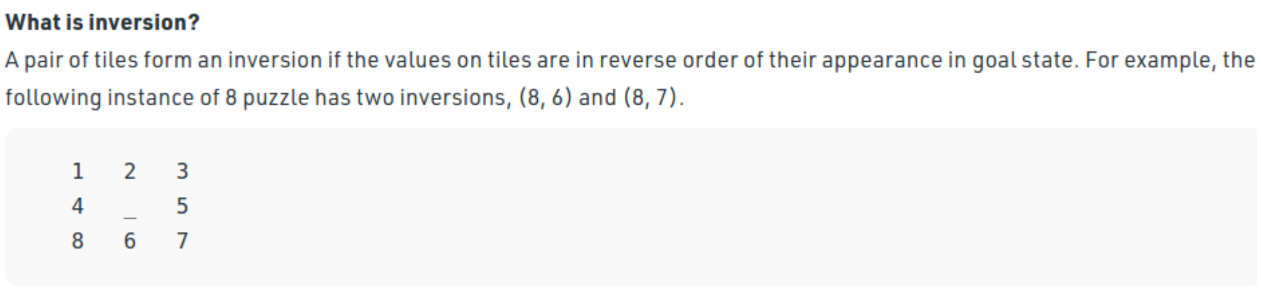
ENDGAME
Checks if there is any valid move left.



```
class EightPuzzle : public BoardGame2D
{
public:
    EightPuzzle();

    void playUser(string move); /*playUser takes a string as a parameterand plays the game accordingly */
    void playAuto(); /* playAuto plays the game by the computerfor one move. */
    void print() const;  /* prints the game on the screen starting from the top left corner of the terminal. *
    bool endGame(); /* endGame returns true if the game is ended. */
    bool check_input(string userInput, int gameType);
    bool check_move(string move, int gameType);
    int boardScore(); /* boardScore returns an intscore value for the current board */
    void initialize(); /* initializes the board.  */

private:
    vector<vector<int>> board;
};
```

VALID MOVE TYPE FOR EIGHT PUZZLE IS:
2U, 3D, 5L, 2R
2 is the number, U,R,D,L is the direction

WHAT I RETURN FROM BOARDSCORE IN EIGHT PUZZLE IS:
It counts the inversion number of the board. If the game has finished it returns 0.



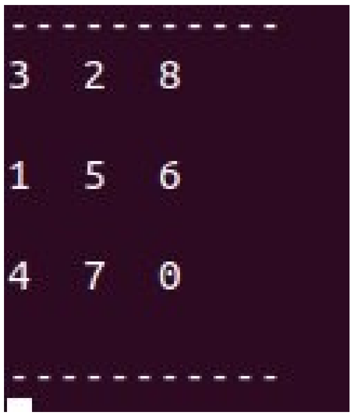
What is inversion?

A pair of tiles form an inversion if the values on tiles are in reverse order of their appearance in goal state. For example, the following instance of 8 puzzle has two inversions, (8, 6) and (8, 7).

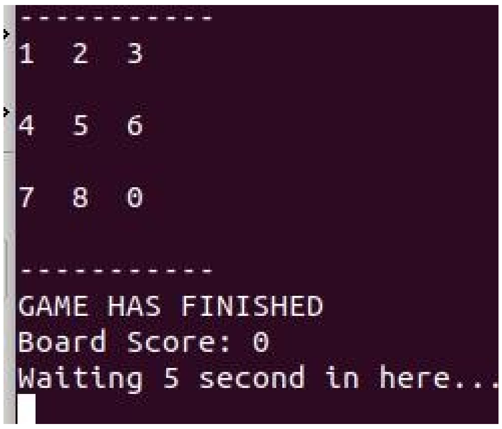
```
1   2   3
4   _   5
8   6   7
```

When initializing the board, I am counting the inversion every time board created, if inversion is divided by 2, that means that board is solvable, if not that means that board is not solvable.

ENDGAME
Checks if the inversion number is 0 or not. If it is 0, then it means our board is ordered well.

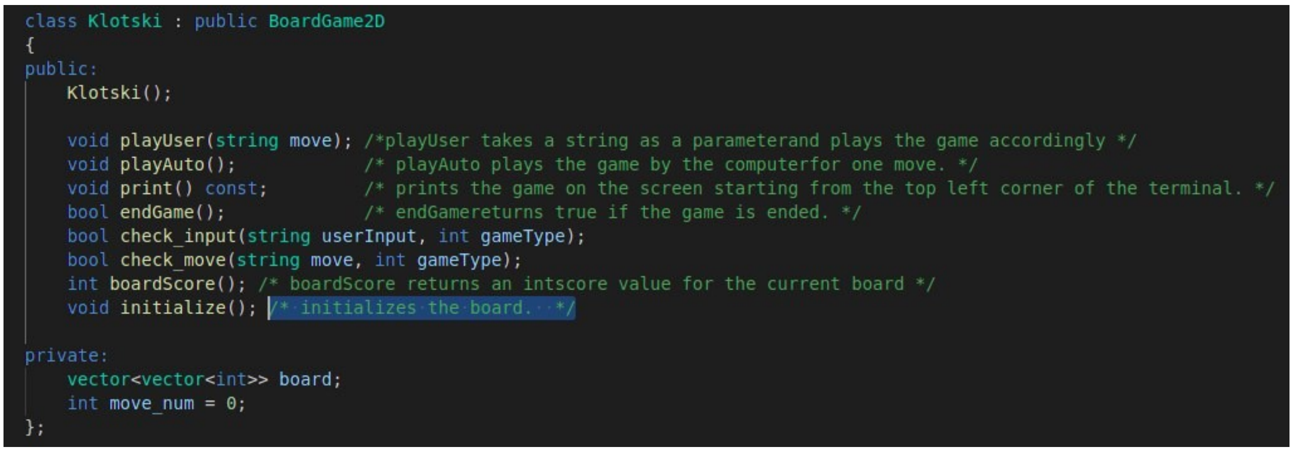


```
-----------
3   2   8

1   5   6

4   7   0

-----------
```



```
-----------
1   2   3

4   5   6

7   8   0

-----------
GAME HAS FINISHED
Board Score: 0
Waiting 5 second in here...
```

BEGINNING   END

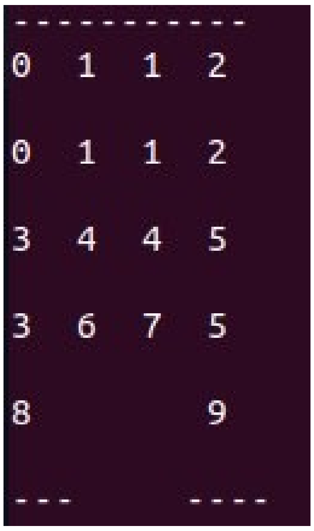


```
class Klotski : public BoardGame2D
{
public:
    Klotski();

    void playUser(string move); /*playUser takes a string as a parameterand plays the game accordingly */
    void playAuto();            /* playAuto plays the game by the computerfor one move. */
    void print() const;         /* prints the game on the screen starting from the top left corner of the terminal. */
    bool endGame();             /* endGamereturns true if the game is ended. */
    bool check_input(string userInput, int gameType);
    bool check_move(string move, int gameType);
    int boardScore(); /* boardScore returns an intscore value for the current board */
    void initialize(); /* initializes the board.  */

private:
    vector<vector<int>> board;
    int move_num = 0;
};
```

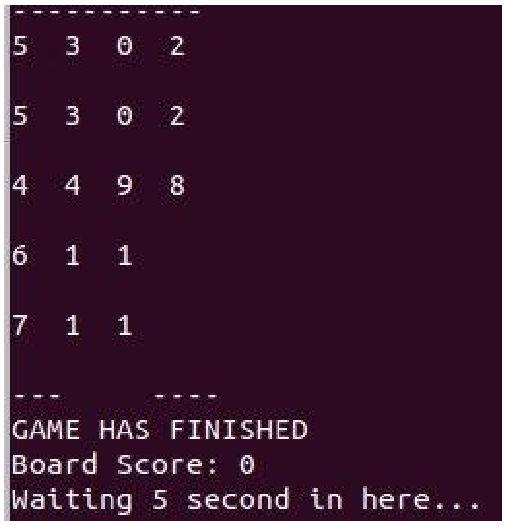
VALID MOVE TYPE FOR KLOTSKI IS:
2U, 3D, 5L, 2R
2 is the number that the shape has, U,R,D,L is the direction

WHAT I RETURN FROM BOARDSCORE IN KLOTSKI IS:
It counts the number of moves and returns it.
If the game has finished it returns 0.

ENDGAME
Checks if square is in the at the at the bottom of the center.



```
-----------
0  1  1  2

0  1  1  2

3  4  4  5

3  6  7  5

8        9

---      ----
```



```
-----------
5  3  0  2

5  3  0  2

4  4  9  8

6  1  1

7  1  1

---     ----
GAME HAS FINISHED
Board Score: 0
Waiting 5 second in here...
```

BEGINNING   END

I put delay before the print statements. For the peg solitiare it does not be a problem but for the other 2 games, sometimes it can take 20 seconds to solve the board.
So I gave their sleep time very very small.

For the Peg Solitiare: `sleep_until(system_clock::now() + milliseconds(150));`
For Klotski : `sleep_until(system_clock::now() + nanoseconds(5));`

For Eight Puzzle : `sleep_until(system_clock::now() + nanoseconds(10));`