Chess Program 5 - Extract a `Piece` class from the `Chess` class.

1. Modify `main()` to have the chess game instance as a pointer stored by reference (on the Heap), rather than by value (on the Stack). Rather than having `Chess chessGame = Chess();` you will have `Chess* ptrChessGame = new Chess();`
2. Notice that you now must explicitly `delete` the `ptrChessGame`, otherwise you will have a memory leak!

Separate the concerns of the `Chess` class so to use a new class called `Piece`. See over for the structure of your `Piece` and `Chess` classes.

1. Your `Piece` class should maintain the `mRow`, `mColumn`, and `mType` private variables. It's constructor should accept `row`, `column`, and `type` variables. The constructor should look like so, `Piece(int row, int column, char type)`.
2. Add `getRow()`, `getColumn()`, and `getType()` accessors to `Piece`.
3. It's useful to add a `print()` method to `Piece` for inspecting your `Piece` instance.
4. Use Preprocessor commands to add the two new files `piece.hpp` and `piece.cpp` for your `Piece` class. Notice Preprocessor linkage gets more complex with more classes.

Have your `Chess` class create a `Piece` instance.

1. Your `Chess` class will create, maintain, and destroy a `Piece*` pointer (on the Heap) called `mPtrPiece,` stored as a `private` member variable.
2. Add a new method to `Chess` called `makeChessPiece()` that creates the `mPtrPiece`.
3. Modify your `drawBoard()` method to use the accessors to `getRow()`, `getColumn()`, and `getType()` information on your `mPtrPiece` instance.
4. Have your `~Chess()` destructor destroy your `mPtrPiece` instance. Notice that the destructor is useful to clean up references that the class created.
5. Your play() method in Chess should now be very simple, looking like so,

```
void Chess::play()
{
    makeChessPiece();
    drawBoard();
}
```

The structure of your `Piece` and `Chess` classes should look like so,

```
class Piece
{
public:
    Piece(int row, int column, char type);
    ~Piece();

    void print();

    int getRow();
    int getColumn();
    char getType();

private:
    int mRow;
    int mColumn;
    char mType;
};


class Chess
{
public:
    Chess();
    ~Chess();

    void play();
    void makeChessPiece();
    void drawBoard();

private:
    Piece* mPtrPiece;

    int askInt(int min, int max);
    char askType();
    bool isValidType(char pieceType);
};
```