COM S 213 – Fall 2004

Assignment #8

Template Class

Due November 4, 2004

Your assignment this week is to create a *template class* named `Matrix` which takes three placeholders:

1. A data type used to represent each individual element of the matrix
2. An integer representing the number of rows in the matrix
3. An integer representing the number of columns in the matrix

The idea is that the `Matrix` can be used to store a "grid" of data types to be defined at compile time. The Matrix will actually store pointers to the data type used in a two dimensional array. If the element at a particular position in that array is defined (if it has been "placed") the array will contain a pointer to that item. If not, the array will contain NULL. The `Matrix` class will have the following member functions:

- `placeItem` – takes a pointer to the data type as well as an x and y coordinate at which point to place the item. This member function returns a Boolean which indicates if the placement was successful. The placement is considered *unsuccessful* if you attempt to place an item where one already exists.
- `removeItem`—used to remove any item from the grid. Returns a Boolean which indicates if the removal was successful. The removal is unsuccessful if there is no item stored at the specified position.
- `display` – displays the contents of the `Matrix`. It assumes that the data type used in the placeholder for this template class has a member function named `display()` (which displays the item to standard out) and also has a static member function named `displayVoid()` which is used to display some sort of indicator that no item exists at a given location. The logic then becomes that `Matrix::display()` goes through its internal array and either calls display or `displayVoid()` of the placeholder class depending on whether or not a given position is populated.

Finally, you will need to create a simple class to used as the "placeholder" class when it comes time to use the `Matrix` class. This class should have some sort of internal storage to store a name or other identifier which can be displayed when it comes time to display the matrix. An example might be a class named `ChessPiece` which may store the standard abbreviations for each type of chess piece. Again, this class must have two member functions defined:

- `display()` – simply displays the identifier associated with the class
- `displayVoid()` – a static member function used to display something to indicate that there is no piece at a given location.

Your main() function should simply declare an instance of the `Matrix` and populate some spaces with pieces. It should then print out the matrix and free up any dynamically allocated memory.

If you used the idea of having a `ChessPiece` class, and your initial Matrix setup involved setting up a chess board in its standard configuration, your output might look like this:

```
BRk BKn BBi BQn BKg BBi BKn BRk
BPa BPa BPa BPa BPa BPa BPa BPa
 -   -   -   -   -   -   -   -
 -   -   -   -   -   -   -   -
 -   -   -   -   -   -   -   -
 -   -   -   -   -   -   -   -
WPa WPa WPa WPa WPa WPa WPa WPa
WRk WKn WBi WKg WQn WBi WKn WRk
```

Where the abbreviations are as follows:

- BRk – Black Rook
- BKn – Black Knight
- BBi – Black Bishop
- BQn – Black Queen
- BKg – Black King
- BPa – Black Pawn
- WRk – White Rook
- WKn – White Knight
- WBi – White Bishop
- WQn – White Queen
- WKg – White King
- WPa – White Pawn

Please feel free to email me with any questions!