Final Exam- Extra Credit

Extra Credit: Choose one of the two problems below for extra credit. No additional extra credit for implementing both programs. See the 2436 Grading and Submission Guide Sheets for additional grading/submission information. Partial credit will be given. (10 pts)

- Follows same program grading guidelines as Program Sets
- Do not need to show sample runs. Only submit `.cpp` file.

1. Write a C++ program that will let the user enter the size of an array `n` in the range [`3,100`] from the keyboard. The program will then randomly generate `n` integer values in the range [`-100,100`] and lets the user select between sorting the values in ascending or descending order. Also, let the user enter from the keyboard between three different sorts: selection, insertion, and bubble. Convert all letters to uppercase on input. Output to the screen the array after each sorting pass as well as the compares and exchanges after each pass. Once finished output the word: "`Sorted!`" and the total number of comparisons and exchanges needed to sort the list. Use functions for the three different sorts. Finally, the program should ask if the user wants to run the program again (Check case). Refer to the sample output below.

Sample Run:

```
Enter the array size: 6

The 6 random values are: 3 7 5 2 1 6

Enter type of sort [S)election I)nsertion B)ubble]: B

Enter sorting order [A)scending, D)escending]: A

                        Compares Swaps/Exchanges

Pass 1: 3 5 2 1 6 7    |     5         4
Pass 2: 3 2 1 5 6 7    |     5         2
Pass 3: 2 1 3 5 6 7    |     5         2
Pass 4: 1 2 3 5 6 7    |     5         1
Pass 5: 1 2 3 5 6 7    |     5         0

Sorted!

A total of 25 comparisons and 9 exchanges were made.
Array sorted in ascending order by bubble sort: 1 2 3 5 6 7

Run Again (Y/N): N
```

Name the program: `ShowSortsXX.cpp`, where `XX` are your initials.

2. Modify the chess program provided by using appropriate C++ STL classes. Make sure to include all the same functionality and correctly runs in all cases. Include the en passant, castling, and pawn promotion movements in the code. Also, remove any user defined classes, enumerations, and any jumping code

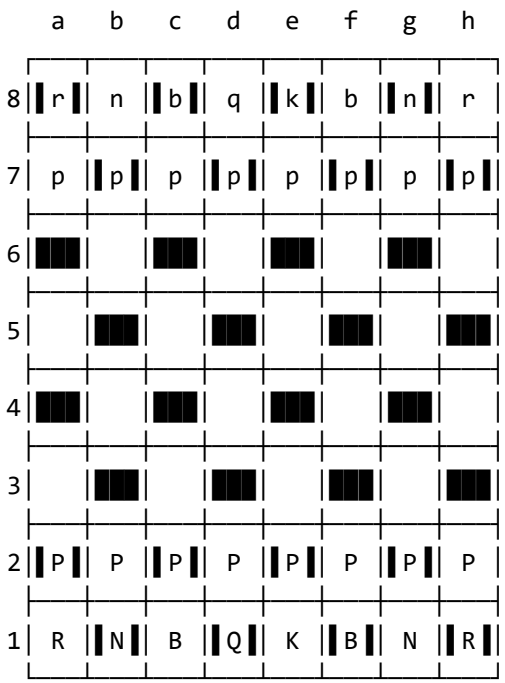
statements. Must use the STL classes for full credit. Output should be user friendly.

- Note full conversion from above must be implemented for full credit. Only reuploading the original file will not get you full credit.

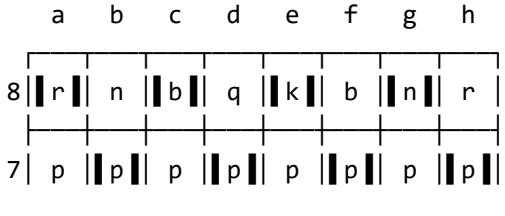
Name the application: `ModifyChessXX.cpp`, where XX are your initials.

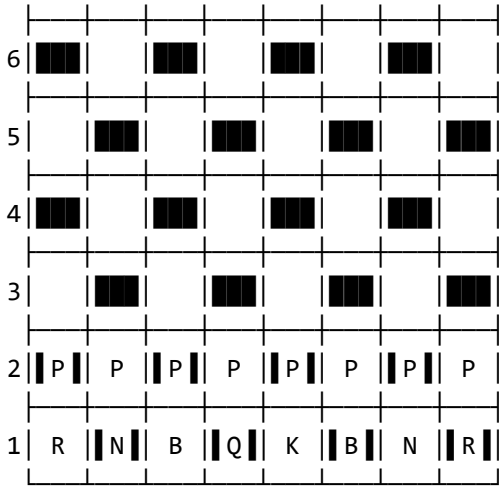
Sample Partial Run

```
Welcome to the game of Chess!

Enter 1 for computer play, 2 for human vs. human: 1
Enter W to play white pieces, or B to play black: W

Computer competitor mode.
Playing white pieces.
Enter ? for help.
```




```
:?

This is a simple game of Chess. You make moves by entering piece coordinates
in the format: "a2 a4", where a2 represents the beginning coordinate of a piece
and a4 represents the desired placement. Normal chess rules apply.
Enter 'Q' by itself to quit the game.
```




6
5
4
3
2 P P P P P P P P
1 R N B Q K B N R

: