Object Oriented Programming & Design
CS 244 - 001
Department of Physics and Computer Science
Medgar Evers College
Exam 3

Instructions:

- **The exam requires completing a set of tasks within 120 minutes.**
- **Write your solutions in the Exam03 directory of your GitHub repository.**
- **Notes are not allowed.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and/or failing to follow any of the rules will result in an automatic zero (0) for the exam.**

TO ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD THE INSTRUCTIONS ABOVE, PRINT YOUR NAME AND THE DATE ON BOTH THIS SHEET AND THE BLUE BOOK

Name: ______________________________ **Date:** ________________

Grading

Section	Maximum Points	Points Earned
1	5	
2	5	
3	5	
4	5	
Total	20	

Section I

In the file `"solutions.txt"` provided write the answers to the following questions

a. What are the names of the getter methods of the *Point* class from the header file `"Components.h"`?

b. What are the names of the setter methods of the *Point* class from the header file `"Components.h"`?

c. What are the names of the getter methods of the *Board* class from the header file `"Board.h"`?

d. What are the names of the setter methods of the *Board* class from the header file `"Board.h"`?

e. What are the names of the getter methods of the *Piece* class from the header file `"Piece.h"`?

f. What are the names of the setter methods of the *Piece* class from the header file `"Piece.h"`?

g. What type of class is *Action* from the header file `"Action.h"`?

h. What are the names of the pure virtual methods of the *Action* class from the header file `"Action.h"`?

i. List the function prototypes of the constructors of the class *Piece* from the header file `"Piece.h"`.

j. What type of class relationship does the classes *Point* and *Piece* have from the header files `"Components.h"` and `"Piece.h"`?

Section II

In the header file named `"PawnPiece.h"` and define the class *Pawn* that must publicly inherit *Action* and contain

- ☐ a public overridden `Move()` method. If *piece* is null, it should return `'\0'`. Otherwise, it should return `'m'` if the line formed from the *Point* of *piece* to the *Point* parameter is a diagonal line, increasing while the team of *piece* equals true or decreasing while the team of *piece* equals false, and length of the line is 1 row. Or it should return `'c'` if the line formed from the *Point* of *piece* to the *Point* parameter is a diagonal line, increasing while the team of *piece* equals true or decreasing while the team of *piece* equals false, and length of the line is 2 rows. Otherwise, it should return the null character (`'\0'`).

  Use the helper functions from the header file `"Components.h"`.

- ☐ a public overridden `ToString()` method. If *piece* is null, it should return the string `""`. Otherwise, it should return the string `"o"` if the team of *piece* is false or the string `"x"` if the team of *piece* is true.

Section III

In the header file named `"KingPiece.h"` and define the class *King* that must publicly inherit *Piece* and contain

- ☐ a public overridden `Move()` method. If *piece* is null, it should return `'\0'`. Otherwise, it should return `'b'` if the line formed from the *Point* of *piece* to the *Point* parameter is a diagonal line. Otherwise, it should return the null character (`'\0'`).

  Use the helper functions from the header file `"Components.h"`.

- ☐ a public overridden `ToString()` method. If *piece* is null, it should return the string `""`. Otherwise, it should return the string `"O"` if the team of *piece* is false or the string `"X"` if the team of *piece* is true.

Section IV

In the header file named `"Player.h"` and define the class *Player* that must contain

- ☐ a private *Piece* array field named *pieces* with a size of 12.
- ☐ a private bool field named *side*.
- ☐ a private copy constructor prototype.
- ☐ a private assignment operator prototype.
- ☐ a public default constructor that assigns false to *side* and an anonymous *Piece* object whose *point*, *team*, and *type* fields are assigned `{0,0}`, *side*, a dynamic *Pawn* object respectively to each of the elements of *pieces*.
- ☐ a public overloaded constructor that takes a bool parameter and assigns the parameter to *side* and an anonymous *Piece* object whose *point*, *team*, and *type* fields are assigned `{0,0}`, *side*, a dynamic *Pawn* object respectively to each of the elements of *pieces*.
- ☐ a public empty destructor.
- ☐ a public *Piece* pointer method named `GetPiece()` that takes no int parameter. If the parameter is between 0 and 11 inclusively, it returns the address of (`&`) the element of *pieces* whose index is equal to the parameter; otherwise, it returns null.
- ☐ a public int constant method named `Actives()` that takes no parameters. It returns the count of the elements of *pieces* that are active (their *active* field is true).