Lab 03 - Inheritance & Polymorphism

Instructions:

- **Your objective is to define a superclass and a subclass in header files named 'Point.h' and 'ChessPoint.h', respectively.**
- **Both header files must contain a header guard.**
- **The classes must be defined within a namespace named 'dsl'.**
- **The header files can only include the libraries *iostream*, *string*, and *sstream*.**
- **Each method excluding special member functions must include pseudocode as a comment above it to receive any credit.**
- **Your submissions must be submitted to the GitHub repository in the Lab03 directory.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating or failing to follow any of the rules above will result in an automatic zero (0) for the lab.**

Grading

Task	Maximum Points	Points Earned
1	3	
2	2	
Total	5	

Note: solutions will be provided for tasks colored blue only.

Task 1

- Within the namespace *dsl* of the file '`Point.h`', define the class *Point* that contains
  - ☐ A private int field named *horizontal*.
  - ☐ A private int field named *vertical*.
  - ☐ A public default constructor that assigns 0 to both fields.
  - ☐ A public overloaded constructor that takes two integer parameters. It assigns the first and second parameters to *horizontal* and *vertical*, respectively.
  - ☐ A public empty destructor.
  - ☐ A public overloaded assignment operator.
  - ☐ A public int constant method named `x()` that takes no parameters and returns *horizontal*.
  - ☐ A public int constant method named `y()` that takes no parameters and returns *vertical*.
  - ☐ A public virtual *Point* reference method named `set()` that takes two integer parameters, assigns the first and second parameters to *horizontal* and *vertical*, respectively, and returns the deference this pointer.
  - ☐ A public virtual *Point* reference method named `shift()` that takes two integer parameters, adds the first and second parameters to *horizontal* and *vertical*, respectively, and returns the deference this pointer.
  - ☐ A public virtual string constant method named `toString()` that takes no parameters and returns a string in the format

    "(*x*,*y*)"

    where *x* and *y* are the values of *horizontal* and *vertical*, respectively.
  - ☐ A friend overloaded ostream operator with a display in the same format as `toString()`.
  - ☐ A friend overloaded equal operator (`==`) that takes two constant *Point* reference parameters. It returns true if the parameters' fields are equal; otherwise, it returns false.
  - ☐ A friend overloaded not equal operator (`!=`) that takes two constant *Point* reference parameters. It returns true if any of the parameters' fields are not equal; otherwise, it returns false.

Task 2

- Within the namespace *dsl* of the file '`ChessPoint.h`', define the class *ChessPoint* that publicly inherits *Point* and contains
  - ☐ A public default constructor that assigns 0 to both fields.
  - ☐ A public overloaded constructor that takes two integer parameters. It assigns the first and second parameters to *horizontal* and *vertical*, respectively if both parameters are in the range [0,7]; otherwise, it assigns 0 to both fields.
  - ☐ A public empty destructor.
  - ☐ A public overloaded assignment operator.
  - ☐ A public overridden `set()` method with return type *ChessPoint* reference that assigns the first and second parameters to *horizontal* and *vertical*, respectively, only if both parameters are in the range [0,7], and returns the deference this pointer.
  - ☐ A public overridden `shift()` method with return type *ChessPoint* reference that adds the first and second parameters to *horizontal* and *vertical*, respectively, only if both sums are in the range [0,7], and returns the deference this pointer.
  - ☐ A public overridden `toString()` method that returns a string in the format

    "[*x*:*y*]"

    where *x* and *y* are *horizontal* represented as the corresponding lowercase letter in the range [a,h] and the value of *vertical* plus 1, respectively.

Extra Credit

- Create a cpp file named '`extra.cpp`' that defines
  - ☐ a Boolean function named `BishopMove()` that takes two constant *ChessPoint* reference parameters and returns true if the second parameter is a valid bishop move from the first parameter; otherwise, it returns false.
  - ☐ a Boolean function named `RookMove()` that takes two constant *ChessPoint* reference parameters and returns true if the second parameter is a valid rook move from the first parameter; otherwise, it returns false.
  - ☐ a Boolean function named `KingMove()` that takes two constant *ChessPoint* reference parameters and returns true if the second parameter is a valid king move from the first parameter; otherwise, it returns false.
  - ☐ a Boolean function named `QueenMove()` that takes two constant *ChessPoint* reference parameters and returns true if the second parameter is a valid queen move from the first parameter; otherwise, it returns false.
  - ☐ a Boolean function named `KnightMove()` that takes two constant *ChessPoint* reference parameters and returns true if the second parameter is a valid knight move from the first parameter; otherwise, it returns false.