Chess Program 8 - Inheritance Part B, Polymorphism.

1. Revert back to having your `mPtrPiece` in Chess to being a pointer of type `Piece*` instead of a pointer to type `Rook*`, but still `new` the piece up as a `Rook`. Notice this does not cause a compiler error. Rook extends Piece, so it's acceptable to have `Piece* mPtrPiece` point to a `Rook` instance. BUT, now at execution the `getSymbolAt()` method in `Piece` gets called instead of `getSymbolAt()` in `Rook`.
2. To get the compiler to call `getSymbolAt()` in `Rook` instead of the one in `Piece` simply declare `getSymbolAt()` in the `Piece` header as `virtual`. e.g.
   ```
   virtual char getSymbolAt(int row, int column);
   ```
3. This is an example of Polymorphism. If a method is declared `virtual` the compiler checks for overriding methods in extending classes and calls those methods instead of the virtual method.
4. Go ahead and create extending classes `Knight`, `Bishop`, `Queen`, and `King` based on what you did for `Rook`.
5. In `makeChessPiece()` use a `switch()` statement to assign `mPtrPiece` to the appropriate class after asking the user for the piece type, e.g.,
   ```
   switch (type)
   {
       case ROOK:
           mPtrPiece = new Rook(row, column, type);
           break;
       case KNIGHT:
           mPtrPiece = new Knight(row, column, type);
           break;
       case BISHOP:
           mPtrPiece = new Bishop(row, column, type);
           break;
       case QUEEN:
           mPtrPiece = new Queen(row, column, type);
           break;
       case KING:
           mPtrPiece = new King(row, column, type);
           break;
   }
   ```
1. You will need in include the headers for Rook, Knight, Bishop, Queen, King in `chess.cpp`.
2. Your `virtual` method `getSymbolAt()` in `Piece` should now never actually be called, nor logic in it used. Remove the old logic and change it to this,
   ```
   char Piece::getSymbolAt(int indexRow, int indexColumn)
   {
       return ' ';
   }
   ```
3. Your Chess game is now fully using inherited classes and polymorphism.