1. A high-level description of each of your public member functions in each of your classes, and why you chose to define each member function in its host class; also explain why (or why not) you decided to make each function virtual or pure virtual. For example, "I chose to define a pure virtual version of the sing() function in my base Piece class because all Pieces sing, but each kind of piece sings in its own way."

* Class Well
  + **Well** – Constructor
  + **~Well** – Destructor (unnecessary)
  + **emptyWell** – this function is to restart the tetris board once a level is complete, it clears the board and allows for the new game to begin. I placed this in Well class because it dealt directly with the well object
  + **updateTet** – updates the coordinate with a given character, placed in the Well class because it directly modified the well behavior.
  + **viewTet** – This function allows the user to view the coordinate of the tetris board without modifying it, which is why it is constant
  + **floors** – deals with the most important part of the game, removing of completed floors. It tracks how many floors have been filled and returns the amount to be removed.
* Class Game
  + **Game** – Constructor, initialized almost everything
  + **~Game** – for pointer destruction
  + **Play** - work horse of the game. Starts a timer which is what the entire function is revolving around, if a certain time elapses from a moment when time is measured, a piece drops. This function waits on key inputs and is in charge of updating variables revolving around playing individual levels.
  + **playOneLevel** – rather unnecessary, behaves as a helper function. Returns true if the player has removed the required level count.
  + **DisplayPrompt** – instructs user on what to do next, displays string
  + **displayStatus** – displays level, rows left, and score
  + **getRandPiece** – assigns a random piece type to a pointer
* Class Piece & Children
  + **Piece –** initializer for drop location and orientation
  + **pDisplay –** made virtual because each child displays something different
  + **PieceInWell –** places a tetris piece into the game at X=3, Y=0, with default orientation.
  + **nextPieceDisplay –** used only for the display portion of the game, prints the next tetris piece in line.
  + **Rotate –** changes the orientation of the tetris piece
  + **pieceFall –** moves the tetris piece down one coordinate if it is allowable. This function determines if it can move into the position or not.
  + **moveLeft –** moves the tetris piece to the left, if allowable. Virtual because the crazy piece moves to the right
  + **moveRight –** moves the tetris piece to right, if allowable. Virtual because the crazy piece moves to left.
  + **placePiece –** virtual function because of the foam piece and the vapor piece. Otherwise it turns the tetris piece into bricks once it has reached a lower bound.

1. A list of all functionality that you failed to finish as well as known bugs in your classes, e.g. "My VaporBomb doesn't work correctly, so for now I just treat it like a normal piece." or "I couldn't get the recursive FoamBomb algorithm to work, so for now it just checks its four immediate neighbors to see if they can be foamed."
   * I could not get the foam piece to work correctly. It forms like it should, my recursive algorithm is faulty though. Instead of making a square around it, it shoots a rhombus directly above it. I couldn’t fix this in time. I spent too much time reworking this project. I spent over 60 hours and restarted 5 times. The most difficult part was getting the pieces to display. Lots of trial and error when messing with the classes and the member functions as well as what should go where and what should be called by which function. Super confusing.
2. A list of other design decisions and assumptions you made (e.g., It was ambiguous what to do in situation X, so this is what I decided to do.)
   * I assumed that the tetris board cannot be changed. It will remain the dimensions specified in the spec. I used vectors for my first 3 attempts. But I bit off more than I could chew and made my code more complicated than it had to be. I’m pretty happy with the final result though. Given another couple of hours. I think I could have worked out the foam bomb and any potential bugs I didn’t catch. I also assumed the play another level function was simply a helper function. It was kind of redundant to use it when a simple if statement did the same thing. Testing it was time consuming because I had to play tetris.
   * Side Note. I had an interesting bug where the right most of my well wasn’t being cleared. I didn’t realize because when I play tetris. The right most column is where I place my I pieces for clearing the board. I fixed it.