1. **A description of the control flow for the interaction of a player avatar and a bank square. Where in the code is the co-location of the two objects detected, and what happens from that point until the interaction is finished? Which functions of which objects are called and what do they do during the handling of this situation?**
   * On line 348 of Actor.cpp, a *PlayerAvatar* (the class for peach and yoshi) object moves 2 pixels forward in its *PlayerAvatar::doSomething()* function when its *getActivityStatus()* (line 32 of Actor.cpp) function returns that it is walking (*STATUS\_WALKING)*. Afterward, the *Square::doSomething()* function is called on line 712 of Actor.cpp for ALL Square-type objects (Coin, Directional, Star, Bank, Event, and Dropping Squares), calling its subclasses *BankSquare::doSpecialActivity()* function with the parameter of a pointer to one of the players (it calls once for both players). This pointer is obtained by calling *Actor::getWorld()*, located on line 22 of Actor.cpp, which returns a pointer to the *StudentWorld* object that each *Actor* resides in, and then *getPeach()/ getYoshi()* on the obtained object (lines 170 and 175 of StudentWorld.cpp respectively). *BankSquare*’s *doSpecialActivity()* function is on line 790 of Actor.cpp. First, it checks if the *player* is stationary by calling its *PlayerAvatar::getStationaryTicks()* function (line 487 of Actor.cpp). This verifies that the *player* is stopped on a square and has remained there for exactly 1 tick. If true, it checks whether the *player* landed on or moved over the Square using the *Actor::landOn\_moveOver()* function (line 37 of Actor.cpp). This function takes a *PlayerAvatar* pointer and another function as a parameter. The *player* pointer is used to verify that its x-coordinate and y-coordinate (using the provided *getX()* and *getY()* functions) are equal to the current object’s coordinates and also checks if the *player’s getRollNum()* is equal to or greater than 0 using the function in the parameter (these helper functions are located on line 5 and 9 of Actor.cpp). The helper functions to distinguish between a *player* landing on and moving over the Square.
     + If the *player* landed on the *BankSquare* and has remained there for 1 tick, the *BankSquare::giveAllCoins()* function is called (line 801 of Actor.cpp) which gets the *player*’s coins (using *getCoins()* on line 452 of Actor.cpp) and adds it to the current bank balance (using *getWorld()* and calling *getBankBalance()* on it, found on line 256 of StudentWorld.cpp). The *player*’s coin amount is set to that amount (using the *setCoins()* function on line 457 of Actor.cpp), the bank balance is cleared (using *setBankBalance()* called on *getWorld()*, found on line 261 of StudentWorld.cpp), and the *SOUND\_WITHDRAW\_BANK* sound is played using the provided *playSound()* function.
     + If the *player* moved on the *BankSquare*, the *BankSquare::takeSomeCoins()* function is called (line 809 of Actor.cpp) which takes up to 5 coins from the player and deposits it in the central bank. First we get the number of coins from the *player* (*getCoins()*) and the current bank balance (*getWorld()->getBankBalance()*). If the player has less than *BANK\_DEDUCTION\_AMOUNT* (5 coins), we set their coin amount (*setCoins()*) to 0 and increase the central bank account by all of the *player*’s coins (*getWorld()->setBankBalance()*). Otherwise, subtract 5 coins from the player and deposit 5 coins to the central bank. Finally, play the *SOUND\_DEPOSIT\_BANK*.
   * The interaction is now complete.
2. **A list of all functionality that you failed to finish as well as known bugs in your classes**
   * None
3. **A list of assumptions you made**
   * For the *CoinSquare* class, I did not make two separate subclasses for coins that take and give 3 coins. Instead, a blue coin would pass in “3” as their *m\_grant\_amount* and a red coin would pass in “-3” as their *m\_grant\_amount*. The class itself doesn’t distinguish between either coin and instead affects the player’s coin amount by their grant amount.