Adam Quintana
CS 6015
Assignment 17
Refactoring the simulation program

1. **Eliminate Global Variables**
   I was initially storing a global variable *int runningClock* to track time the time globally. It was working in this manner, however, I could tell that it would be susceptible to bugs down the road if I ever refactored the code. Therefore, I removed it as a globale variable and now my runSimulation() functions initializes it and passes it by reference to any functions that need it.

2. **Duplicate code**
   My 'arrive()' function supports both the bank scenario and the supermarket scenario through an if-else branch. However, the following code was being run at the start of both branches. I extracted it out to run before either branch and removed the duplicate code.

```
//find minimum projected finish and queue index
std::tuple<int, int> minQueueTuple = findMinimumProjectedFinishAndIndex(queues, runningClock);
int minQueueProjectedFinishTime = std::get<0>(minQueueTuple);
int minQueueIndex = std::get<1>(minQueueTuple);
assert(minQueueProjectedFinishTime >= *runningClock);
assert(minQueueIndex >= 0 && minQueueIndex <= NUMBER_OF_CASHIERS);
```

3. **Code organization**
   This wasn't part of the 'Reasons to Refactor' checklist in the textbook, but I felt it was important enough to include. Originally, I had all my classes, structs, enums and functions under Main.cpp. This proved to be challenging when I went to find a particular part of the code.

   Old:
   Main.cpp – contains EVERYTHING (~356 lines)

   New:
   Main.cpp – contains main() and main functions for the simulation (~220 lines)
   Customer.cpp/.hpp – contains the customer class and methods (~40 lines)
   Event.cpp/.hpp – contains the event class and methods (~80 lines)
   Utilities.cpp/.hpp – contains functions for printing and percentile calculations (~40 lines)

   I committed and tested the code before I made each of these changes. I then tested my code after and it still worked as expected.