Project outline:

The purpose of this document is to outline all of the constants, methods, and classes within our model so that they can be worked on asynchronously.

Constants:

primary list: numpy array of items representing the pool of primary items customers can have

secondary list: numpy array of items representing the pool of secondary items customers can have

full list: numpy array of items representing the pool of all items customers can have

view range: int representing the distance in any direction a customer can purchase items from

delta customer: int representing the number of steps between customers entering the store

total customers: int representing the total number of customers which can enter the store

store size: int representing the size of the array which will represent the store (the store will be square)

max time: int representing the number of steps which can pass before forcibly ending the simulation

number primary list: int representing the number of items a customer will have in their primary list

number secondary list: int representing the number of items a customer will have in their secondary list

global return values:

customer steps: int representing the number of steps made in the current simulation

items sold: int representing the number of items sold in the current simulation

money made: float representing the amount of money made in the current simulation

Methods:

initItems: takes a numpy array of strings and a numpy array of floats, which paired together represent an item. Returns a numpy array of all items created from this.

createStore: takes a 2D numpy array of coordinates (a[0,:] = x coords. A[1,:] = y coords), and returns a numpy array of shelves created in each coordinate, with a random item per shelf.

createCustomer: takes a numpy array of primary items, a numpy array of secondary items, a numpy array of floats representing percentages in an item density for primary items (defaults to None if unused), and a numpy array of floats representing percentages in an item density for secondary items (defaults to None if unused). Returns a single Customer created from this information, with primary and secondary items chosen based off of random selection (with the probability density if provided) up to the number specified in the constants.

createCustomerList: takes an integer for the number of customers to create, a numpy array of floats representing percentages in an item density for primary items (defaults to None if unused), and a numpy array of floats representing percentages in an item density for secondary items (defaults to None if unused). returns a numpy array of that many Customers, using the densities if given.

moveCustomer: takes a customer, a numpy array representing the shelves of the store, and a numpy array of all customers in the store, and calculates the next movement step for the customer, and moves the customer.

customerPurchase: takes a customer and a numpy array representing the shelves of the store, and determines which purchases the customer can make, and removes those items from the customer’s list, and updates return values accordingly

runOneSimulation: takes a numpy array of shelves representing a store, and runs a single simulation by creating a list of shoppers (without a probability density), looping through moving the customers and purchasing items, with customers being placed into the store every set amount of steps. The results are kept in the global result variables.

runOneHundredSimulations: takes a numpy array of shelves representing a store, and runs one hundred single simulations, averaging and returning the results as floats in a numpy array (a[0] = average items sold, a[1] = average money earned, a[2] = average distance walked).

runRandomizedSimulations: runs five sets of one hundred simulations with a different item layout for each set of simulations, outputting the results of each, paired with the item layout of the store.

runAnimatedSimulation: runs a single simulation, and displays all customer positions within the store graphically in each step. Outputs the current totals as the simulation progresses, as well.

runOneSimulationDensity: takes a numpy array of shelves representing a store, a numpy array of floats representing percentages in an item density for primary items, a numpy array of floats representing percentages in an item density for secondary items, and runs a single simulation by creating a list of shoppers (with a probability density), looping through moving the customers and purchasing items, with customers being placed into the store every set amount of steps. The results are kept in the global result variables.

runOneHundredDensitySimulations: takes a numpy array of shelves representing a store, a numpy array of floats representing percentages in an item density for primary items, a numpy array of floats representing percentages in an item density for secondary items. and runs one hundred single simulations, averaging and returning the results as floats in a numpy array (a[0] = average items sold, a[1] = average money earned, a[2] = average distance walked).

Classes:

shelf: class containing a reference to the item which is represented, and a pair of integers representing the position in the store of the item.

Item: class containing a string of which item it is, and a float price for said item.

Customer: class containing a two element numpy array representing the position within the store, a numpy array of items which represents the primary item list, and a numpy array of items which represents the secondary item list.