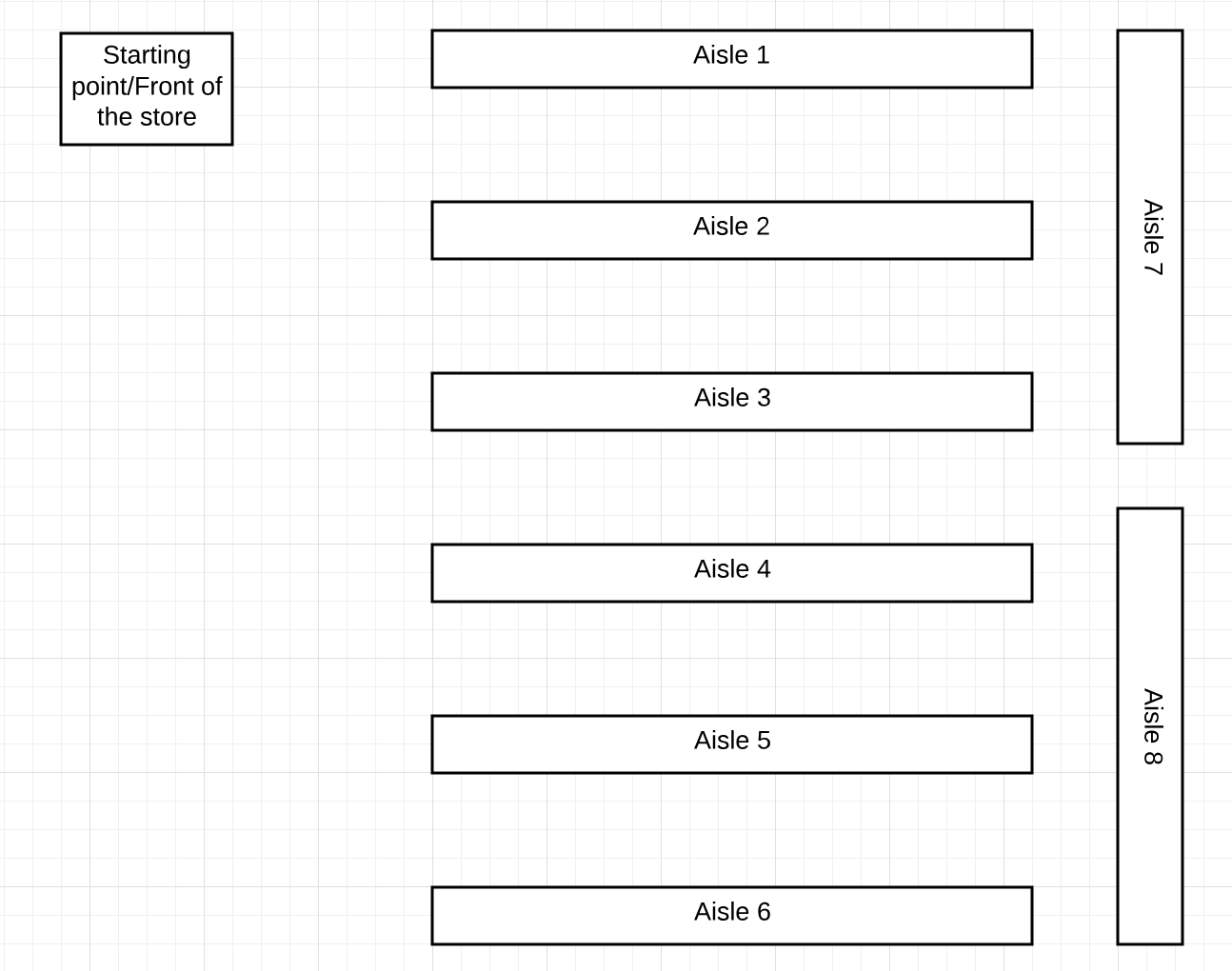
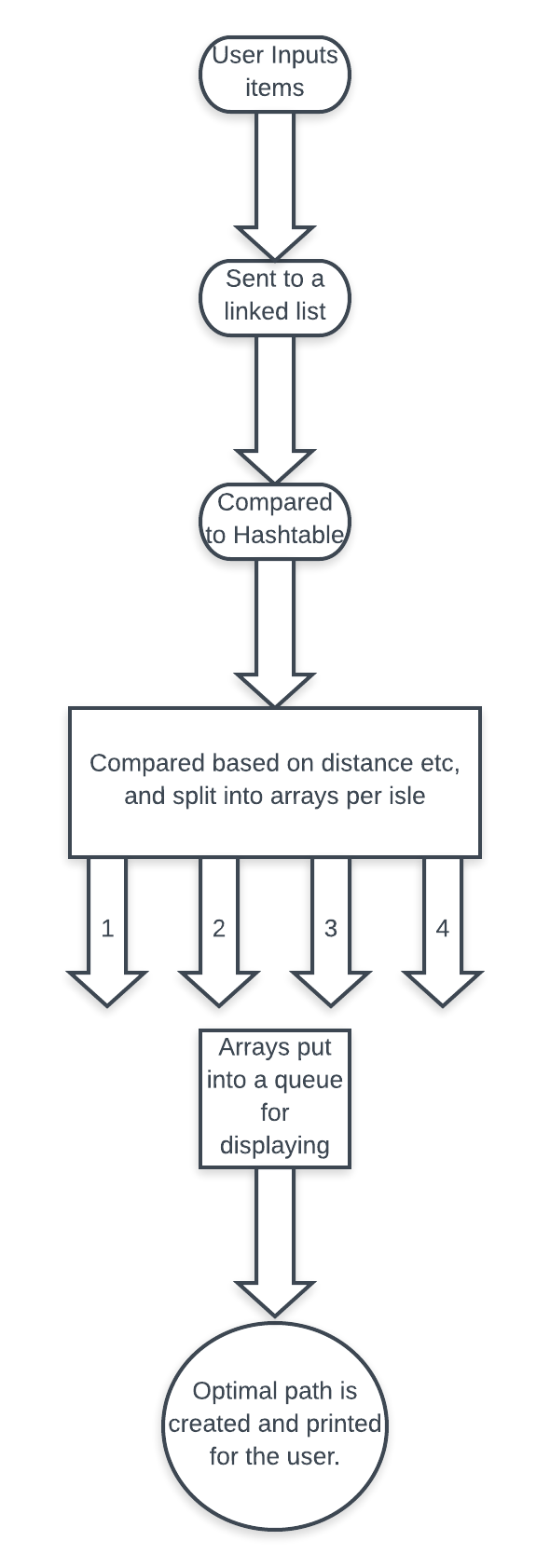
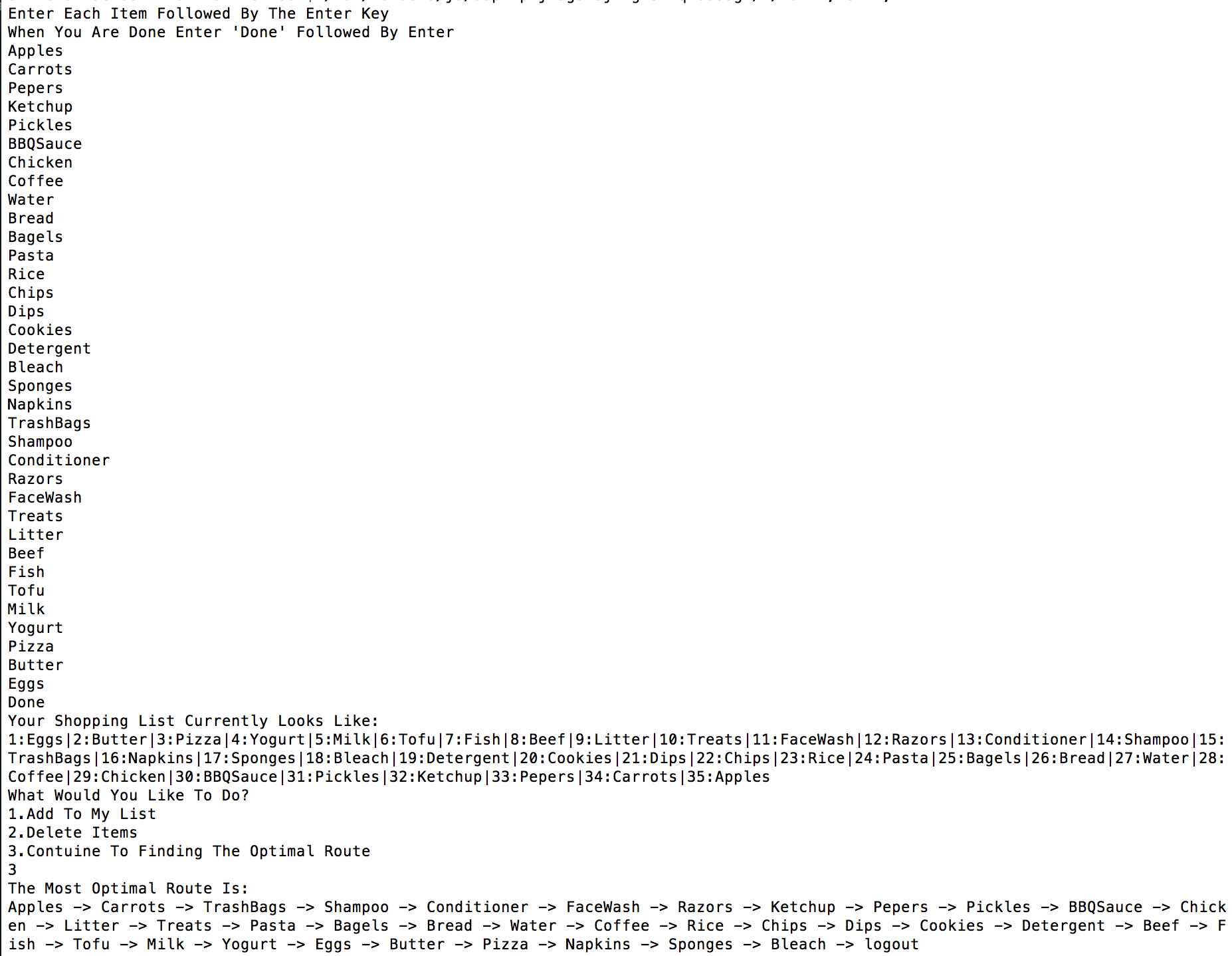
In my project I used Linked Lists, A Hashtable, Arrays, and a Queue. Inorder the linked list was used to obtain user input because of the nature of the data structure it is resizes itself based on input, so that way the user can input as many items as they want to shop for. It is a singly linked list as well making it easy for me to write a program to traverse it from head to tail. There is also functions for the user to edit the linked list, deleting a certain item or adding more items. The linked list is especially effective because there is no order that I need the input to go in so no need for sorting it (That will come later). I used a hashtable for easy look up of the information of the store. In my hashtable I hashed the names of the items and with the items they had values for their asile and the distance the item was from the front of the aisle. That way every time I search a user input from the linked list I can easily obtain all that data from the hash table. After obtaining the data from the hashtable I put everything into a array. There is a array for every aisle in the store. The arrays are string arrays initiated with zeros in each node. This allows me to enter an item in the certain array. Let's say you were searching for an Apple which is an item in the first aisle and the first item from the front you would obtain in outputs Apple, 1, 1. The Apple item would be put into the aisle 1 array in the first position. Another example would be Shampoo, 7, 2. Shampoo would be put into the 7th aisle array int the 2nd position. After searching and storing all the user inputted items into aisle arrays I would then enter each into a queue by pushing the names into a queue. Pushing each array would push the aisle number and the name of the item skipping any unentered nodes. Using my algorithm (Not really I just kinda guessed) It would then show the user the shortest path possible to pick up all their items with zero backtracking.

The Methodology to this project is basically to find the shortest distance possible for a certain path. The basics of the program is that the user enters items they want then the program will take the items and search for them from the hashtable obtaining aisle and distance data then each aisle will be sorted and sent to a queue based the geography of the store.



Above is the methods and the depiction of the store. The sorting right now (might change before the presentation date) is that the store is sorted by sections, so Aisle 1 and the first third of Aisle 7 would be a section, Aisle 2 and the middle third of 7, Aisle 3 and the last third of 7. The user usually starts at the beginning of Aisle 1 then move down the section then starts at the end of Aisle 2, go up to the front of 3 then move down towards the end of Aisle 7. (Its kinda hard to explain email me or check out my presentation on thursday for some fancy graphics. You move down section 1 which should start you at the end of section 2 move up starting at the beginning of section 3 etc.. The program also checks if a Aisle is empty, so if you don’t need an item from Aisle 1 and the first third of Aisle 7 it starts at the front of section 2 instead. A sample input is below



Note: I will have a map available for the presentation but a .txt file is attached with the list of items, their aisle, and their distance relative to the front on the aisle within the zip. Code will be commented as well for clear understanding. Also the sorting seems kinda bad I guess i’ll think of a way to do it better I don’t really have an idea rightnow could have used that thing we learned in class about doing shortest distance but I would have to create a graph of the whole store item by item and factor in walls. Maybe determining the shortest is breaching A.I territory or a bunch of If statements.. For right now I think it's creates one of the shortest paths but definitely not the shortest and fastest. To be honest I just want to show my understand of data structures and how to use them and I hope this project does a decent job at it.