Matt Chupp
mlchupp
CSIS 252
Hash Lab
5/4/15

Problem Summary

The has lab is extending the study of time and data structures to sorting and searching. There will be taking a random batch of unique people and organize them in to different methods. Then another match of people will be taken and looked up in the first structure. The program is going to track the number of compares and swaps for the sort, the number of compares to create the hash table, and for the lookup of both structures, the number of compares.

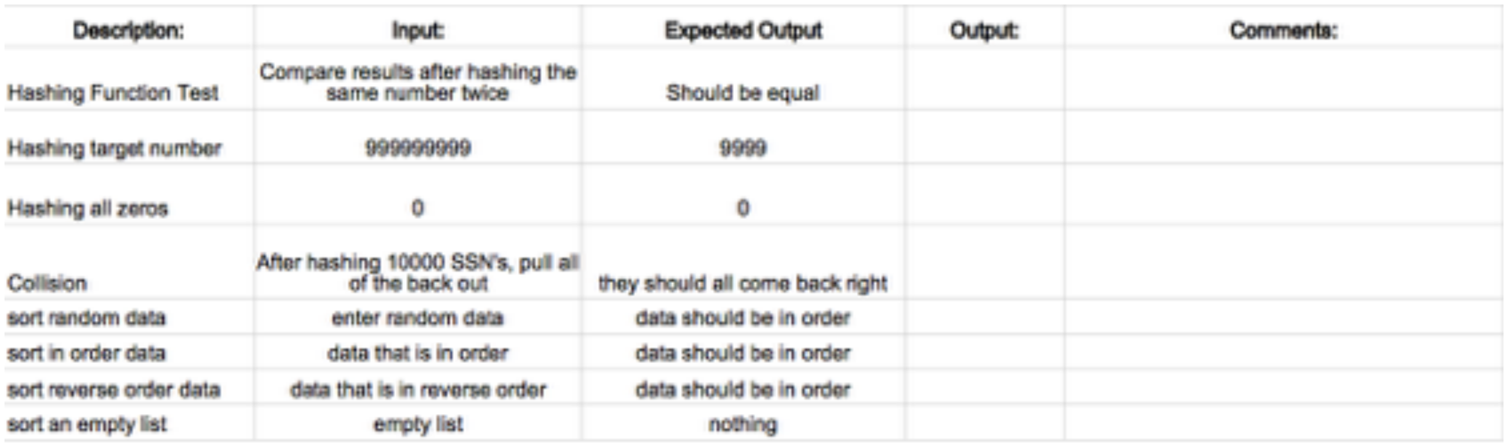
Problem Requirements

- Hash Table
- Sorting
- Data Files
- Using the same interface as the previous labs
- Counting swaps and collisions
- Determine how to handle collisions

System Design — Overview and Detailed Design

[See Page 3]

Testing Report



Description:	Input:	Expected Output	Output:	Comments:
Hashing Function Test	Compare results after hashing the same number twice	Should be equal		
Hashing target number	999999999	9999		
Hashing all zeros	0	0		
Collision	After hashing 10000 SSN's, pull all of the back out	they should all come back right		
sort random data	enter random data	data should be in order		
sort in order data	data that is in order	data should be in order		
sort reverse order data	data that is in reverse order	data should be in order		
sort an empty list	empty list	nothing		

Management Report

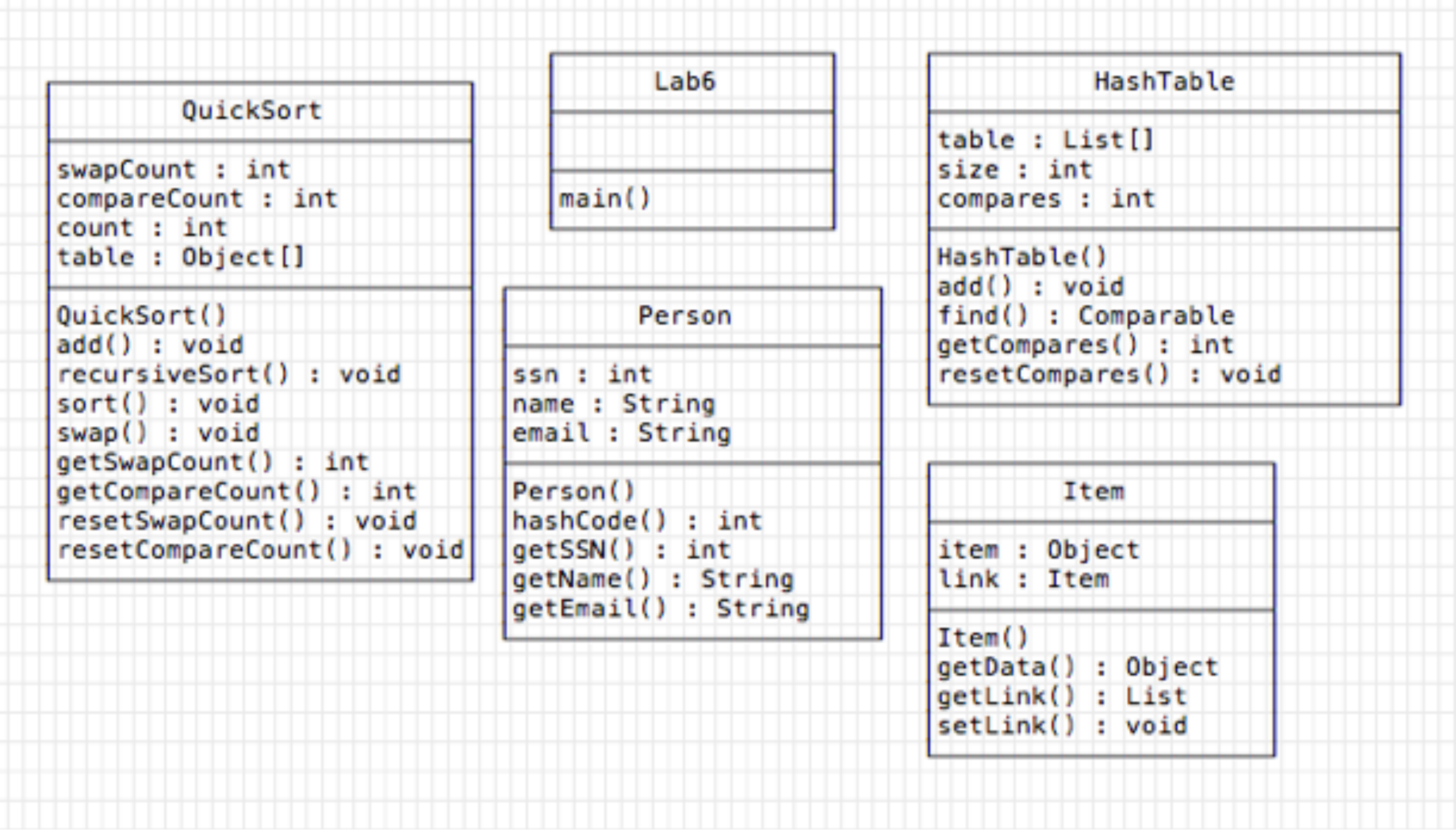
The first part of the SDR, Planning, and UML diagram were completed all within about 2 Hours. The next step is writing all of the code, getting testing data ready, and completing the SDR and documentation. Estimated total time -> 6 hours. The actual completion time was not within budget as writing the Hash Table took longer than expected. Approximately 10 hours were spent total on the project including testing, coding, and the SDR.

Lessons Learned

1. Hash tables and how they work
2. Sorting algorithms and how to design a good algorithm
3. How sorting works in linked lists
4. Collisions and the most efficient way to deal with collisions
5. How to work with large data

Future Improvements

I think the way that I implemented some of my data structures could be changed to be better. Sometimes lists are easier to work with than arrays. Work on my sorting algorithm to make it the most efficient.



QuickSort
swapCount : int
compareCount : int
count : int
table : Object[]
QuickSort()
add() : void
recursiveSort() : void
sort() : void
swap() : void
getSwapCount() : int
getCompareCount() : int
resetSwapCount() : void
resetCompareCount() : void
Lab6
main()
Person
ssn : int
name : String
email : String
Person()
hashCode() : int
getSSN() : int
getName() : String
getEmail() : String
HashTable
table : List[]
size : int
compares : int
HashTable()
add() : void
find() : Comparable
getCompares() : int
resetCompares() : void
Item
item : Object
link : Item
Item()
getData() : Object
getLink() : List
setLink() : void