# Assignment 3

# CSE 503 – Summer 2024


## Richard Douglas

# 1 Assignment Description

We were asked to edit code in a file called hash.cpp containing an incomplete implementation of singly linked list (with a typo). Specifically, we were asked to do:

Starting with the code for Hash Table example (hash.cpp)

1. Write your own main and insert function to insert the sequence of integers { 138, 99, 16, 134, 42, 0, 6, 9, 4, 53, 47, 66) using a table of size 17.
2. Implement your own rehashing algorithm of choice and run the same sequence of input using an initial table of size 7.

# 2 Logic Employed

Hash tables are an interesting data structure utilizing key value pairs. standard implementations exist in many languages, but we implement a custom class here.

In this implementation, we simply use a vector (as is done in the text). Keys can be looked up to find corresponding values. This can be useful. Keys can be made any type, and so can values, depending on the implementation of the hash table. This coding example doesn't really do much to show the utility of a hash table because we are simply asked to insert integer values. The text demonstrates implementations in which strings are hashed by passing each of the char values' corresponding int value into a formula. Since we weren't asked to do that, obviously I didn't…. but that's where a lot of the real magic of hashing comes into play- through hashing the data, we can use the modulo operator to assign the object to some index in the array (vector in this case) we are using to store key value pairs.

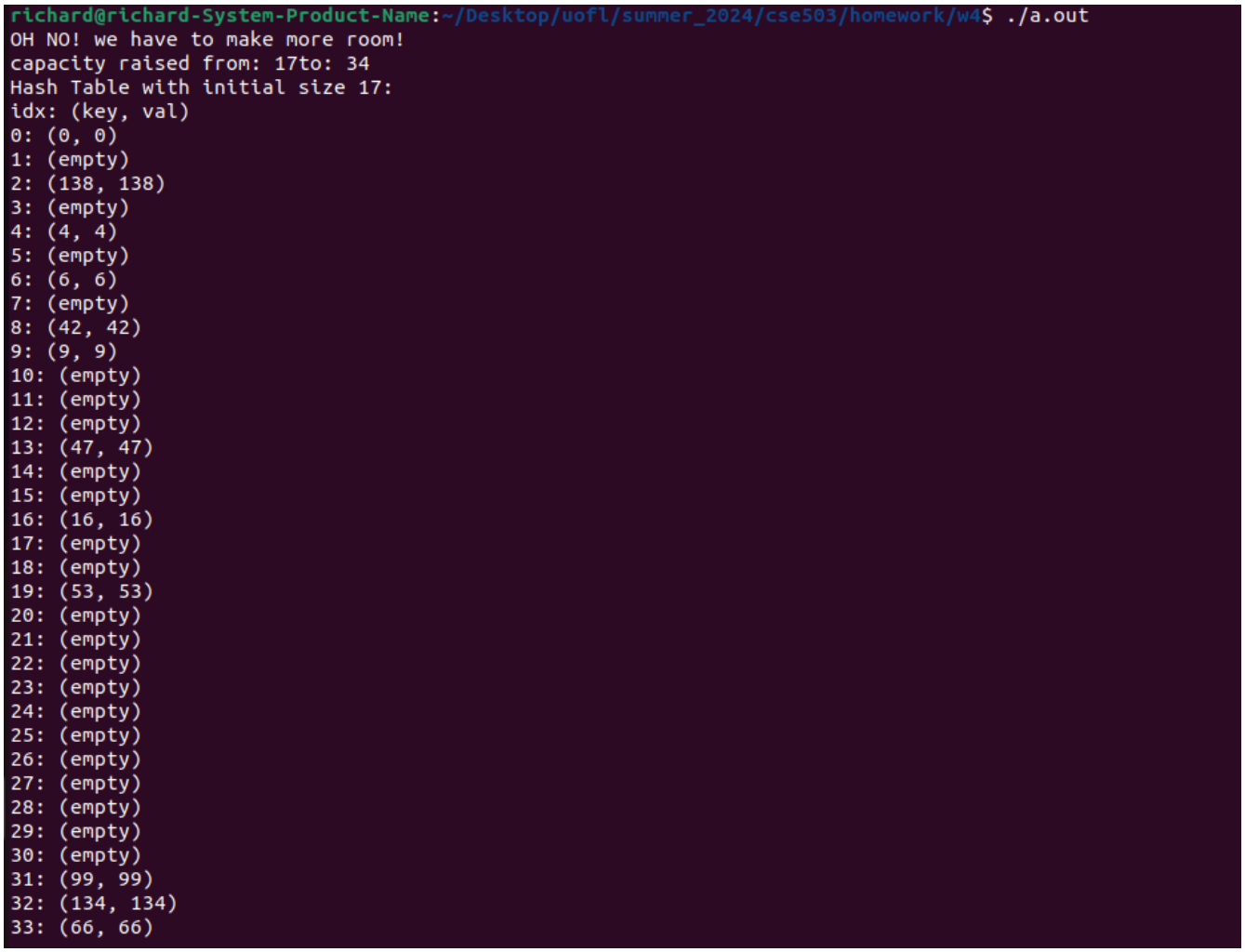
I annotated my code. In previous reports I showed code snippets. In this one, however, I am advising that you may refer to the code for the details of this logic.

# 3 I/O

Our task was again quite simplistic - inserting an array of integer values into a hash table… it again did not show the true power of hashing.

**Screenshots**

The following is the output run in my terminal.



```
OH NO! we have to make more room!
capacity raised from: 7to: 14
OH NO! we have to make more room!
capacity raised from: 14to: 28
Hash Table after rehashing with initial size 7:
idx: (key, val)
0: (0, 0)
1: (empty)
2: (empty)
3: (empty)
4: (4, 4)
5: (empty)
6: (6, 6)
7: (empty)
8: (empty)
9: (9, 9)
10: (66, 66)
11: (empty)
12: (empty)
13: (empty)
14: (42, 42)
15: (99, 99)
16: (16, 16)
17: (empty)
18: (empty)
19: (47, 47)
20: (empty)
21: (empty)
22: (134, 134)
23: (empty)
24: (empty)
25: (53, 53)
26: (138, 138)
27: (empty)
richard@richard-System-Product-Name:~/Desktop/uofl/summer_2024/cse503/homework/w4$
```

## 4 Conclusions

This was an easy assignment. Honestly though, I think it would have been a better exercise and would have showed the power of this data structure if we were at least to hash strings, if not create a class that could handle multiple data types. That seems more college level…. By the way, the source code provided in this course material results in compile errors.