Lab10 Report (all contributions equal)

Andrew McCrary

Vitta Silberberg

Ricky Roberts

Objectives:

1. Hash tables: Hash tables are a data structure useful for storing large amounts of data that do not necessarily need to be ordered but can still be added, found, or removed easily if implemented correctly. This data structure could be useful in a CS career because it has many ways to handle duplicates that are effective and will not overwrite the old input in a spot if implemented certain ways. If a table is filled with objects there may be only 1 value being compared but that does not always mean two entries are entirely the same.
2. Performance: performance is extremely important when it comes to large scale projects or clusters of data. Although our projects can run quickly even with poor implementation and inefficient methods, learning how to write efficient programs can help save time and resources in a CS career for a company, project, etc.

Task 3 running with 3 parts inserted with SKUs of 1, 3, and 5:

A picture containing text

Description automatically generated A picture containing text, electronics

Description automatically generated

Task 4 running with 3 parts inserted with SKUs of 1, 3, and 5:

Text

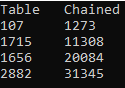
Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

What changes did we need to make to our code for tasks 2 and 3?

1. For task 2, we simply needed to add our Part class to our new project so we could use it as the template type. Then we needed to overwrite the string() conversion operator in order to return the objects SKU as a string for hashing purposes.
2. We kept the basic structure and outline of our testing program from the old lab but made changes to each case that the user chooses. Instead of accessing a linked list, we needed to use an instance of our hash table and its member functions to replicate the functionality of the old lab. Mainly, we changed what object was being used in each case.

Task 5’s Output, left depicts the normal hash table based on an array, and the right depicts a chained hash table.



Discussion about task 5:

On this scale, it appears that the normal hash table requires about 1/10th of the comparisons that the chained hash table requires. This is because the chained hash table requires a comparison for each lesser value, whereas the array supports random access. We believe that these results are as expected and do not require further investigation. The only concern was the spike in comparisons for the hash table of size 150. After looking into it, this appears to happen for each run and will not average out with multiple iterations. We decided to accept that it was flaw within our implementation.

This app has two main methods, a main and an fmain. The normal main is to test the application required in task 5, and the fmain is the general main for testing required in tasks 2-4. Other than that, this application requires no special instructions to run, and should start with a visual studio IDE or .NET compiler.