CSCI 2270 Final Project Report
Kevina Wong

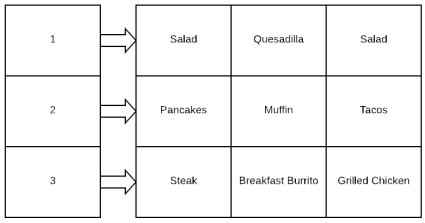
Overview of our project:

Our group decided to make a program that helps student cook and find recipes easier. As college students, we understand how difficult it is to cook with very little time and ingredients. Therefore, we decided to make a program that searches for recipes and prints "best match" recipes to make cooking simpler for students like us.

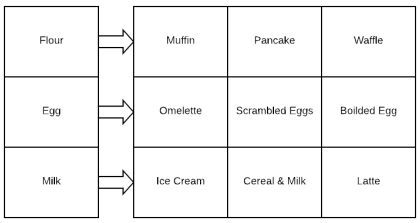
Data Structures (Why&How):

The data structure that we used for this project was a hash table. We decided to use several hash tables within our project. We thought that using hash tables would be the most effective data structures to use because it can be fast without worrying how many recipes we have. Also, searching hash tables would be much faster and more efficient if we were to use a link list or other data structures. Furthermore, the order in which the recipe was placed within the data structure was not particularly important to us, so by using hash tables, we can store a large amount of recipes and access them quickly. We have four hash tables: byName, byIngredient, byMeal, byDiff. These four are all hash tables with the same data within it, but they are arraged differently for our convenience. Here are charts of what each hash table looks like.

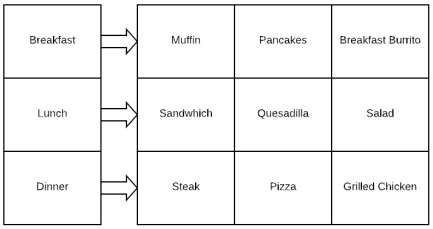
byDiff Hash Table



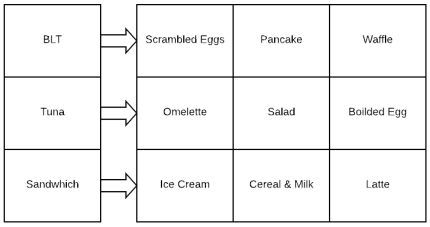
byIngredient Hash Table



byMeal Hash Table



byName Hash Table



**The recipes in the hashtable diagrams are not accurate to what we actually have in our code. They are just examples of how they are set up.

We have four different hash functions to make the searching easier.

byName: We use byName to store all of the recipes in an unordered fashion. This is simply what we use to search for a recipe by name and not by meal type, ingredients, or difficulty.

byIngredient: We use byIngredient to make each hash a “main ingredient”. Each recipe with that main ingredient will be stored in the link list for that hash. This is helpful for when the user wants to search for recipes with the ingredient that they have on hand. Once the user inputs that ingredient, it will print the link list of recipes that chains to the appropriate hash.

byMeal: This hash table is stored in a way that each hash is a meal type (ie. breakfast, lunch, dinner). This is for when the user wants to search recipes by meal type. Once that user input is entered in, the program will print recipes that are associated with that meal type.

byDiff: We have each recipe rated by a difficulty ranking of 1-3 (1 being the hardest and 3 being the easiest). Then, the hash table stores the recipe. Each hash is a difficulty level. Then recipes with that difficulties are chained, so if the user is only looking for easy recipes, that link list will be printed out.

References (if any):

The other data structure that we use is a link list to chain the data to be stored into the hash tables. The reference that I used for this project is just the precious code for hash tables I have done. This includes the starter package for the midterm preparation and assignment 7, an assignment that we have done in the past. I also referred to stack overflow to recall how to write data in at the end of a file for our add recipe menu option.

Link: https://stackoverflow.com/questions/6932409/writing-a-string-to-the-end-of-a-file-c

Result /Sample Outputs: