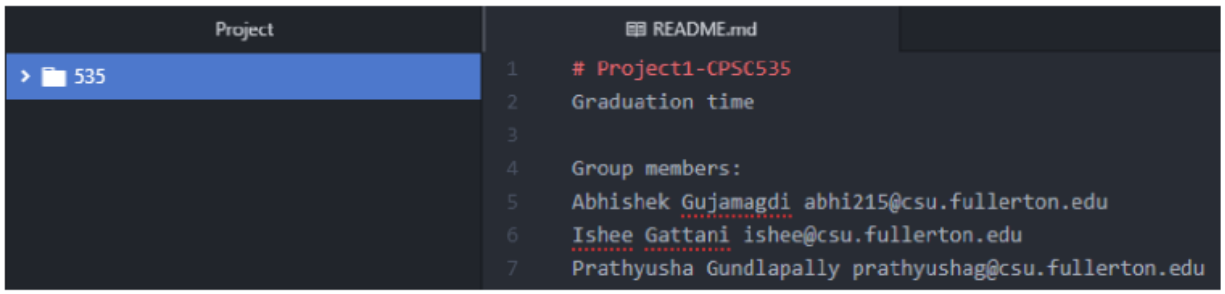
Programming Assignment 2:

Prathyusha Gundlapally: prathyushag@csu.fullerton.edu

Ishee Gattani: ishee@csu.fulleton.edu

Abhishek Gujamagdi: abhi215@csu.fullerton.edu



Pseudo code for overall algorithm:

**Hash function operations **

addItem (key, value):

  index = hashfct(key)

  array[index] = (key, value)

removeItem(key):

  index = hashfct(key)

  array[index] = null

Defining hash functions

int d, barcode;

hashfct(barcode)

d=barcode/10*;

d=d%10;

return d;

//This step is repeated for defining all the other 6 hash functions.

Item t;

t. barcode; t. itemBrand; t. itemColor; t. itemShape are assigned the values.

//Inserting the barcode and item details mentioned above.

hT1[hashfct1(barcode)]. insert ({barcode, t}); // function that adds the specified pair of glasses to main display (i.e., to all hash tables)

hT1.erase(barcode); // function that removes the pair of glasses specified by the barcode from the display

// if pair is found, then it is removed, and the function returns true

// else returns false

bestHashing (); // function that decides the best hash function, i.e. the ones among fct1-fct7 that creates the most balanced hash table for the current ItemCollection data members.

** Calculating the balance of each hash table, one by one**

```
int min = hT1.bucket_size (1), max = hT1.bucket_size (1);
   for (unsigned i = 0; i < 10; ++i)
   {
      if (hT1.bucket_size(i) <= min)
         min = hT1.bucket_size(i);
      if (hT1.bucket_size(i) >= max)
```

```
        max = hT1.bucket_size(i);
    // Call bucket_size () to get the size of each bucket
  }
  balance [0] = max - min;
  min = hT2.bucket_size (1), max = hT2.bucket_size (1);
```

Description of 7 hash functions:

hashfct1(): Hash function to return the hash value based on the **first** digit of the product number.

hashfct2(): Hash function to return the hash value based on the **second** digit of the product number.

hashfct3(): Hash function to return the hash value based on the **third** digit of the product number.

hashfct4(): Hash function to return the hash value based on the **fourth** digit of the product number.

hashfct5(): Hash function to return the hash value based on the **fifth** digit of the product number.

hashfct6(): Hash function to return the hash value based on the **sixth** digit of the product number.

hashfct7(): Hash function to return the hash value based on the **seventh** digit of the product number.

How to run the code:

*.cpp files (Itemcollection.cpp;main.cpp)are the input files that contain the appropriate code for reading a large number of distinct product numbers (7-digit each) and deciding which digit among the seven gives the best balanced storage of the pairs of glasses using hash functions and hash table which can be executed

using Visual studio with a built in option “Run” from the menu bar. Else, Run the command “gcc” (the C-compiler) followed by the full name of your program in the command prompt. This will compile your source code and create an executable file on your desktop.

Steps:

1.User must provide the detailed inputs for each of the above-mentioned Cpp files.

2.Code would be then processed for reading the distinct product numbers first followed by deciding the best-balanced storage.

3. And the output would be printed showing the best hashing function after reading/removing corresponding storage pairs.

Snapshot of code executing the two given input files.