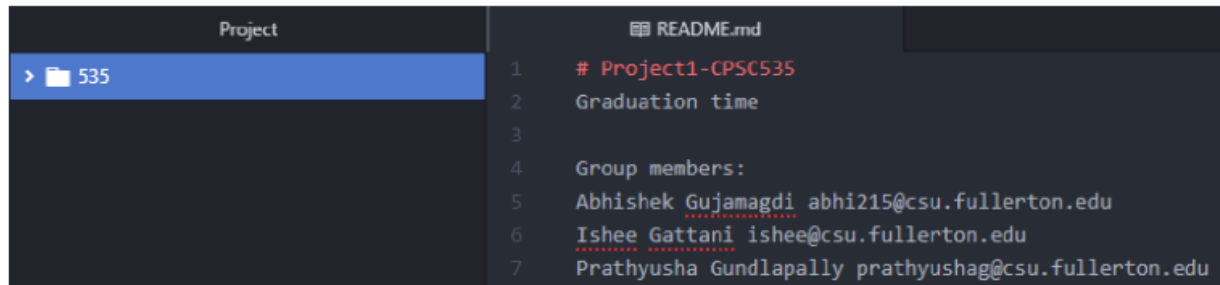


## Programming Assignment 2:

Prathyusha Gundlapally: [prathyushag@csu.fullerton.edu](mailto:prathyushag@csu.fullerton.edu)

Ishee Gattani: [ishee@csu.fullerton.edu](mailto:ishee@csu.fullerton.edu)

Abhishek Gujamagdi: [abhi215@csu.fullerton.edu](mailto: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:

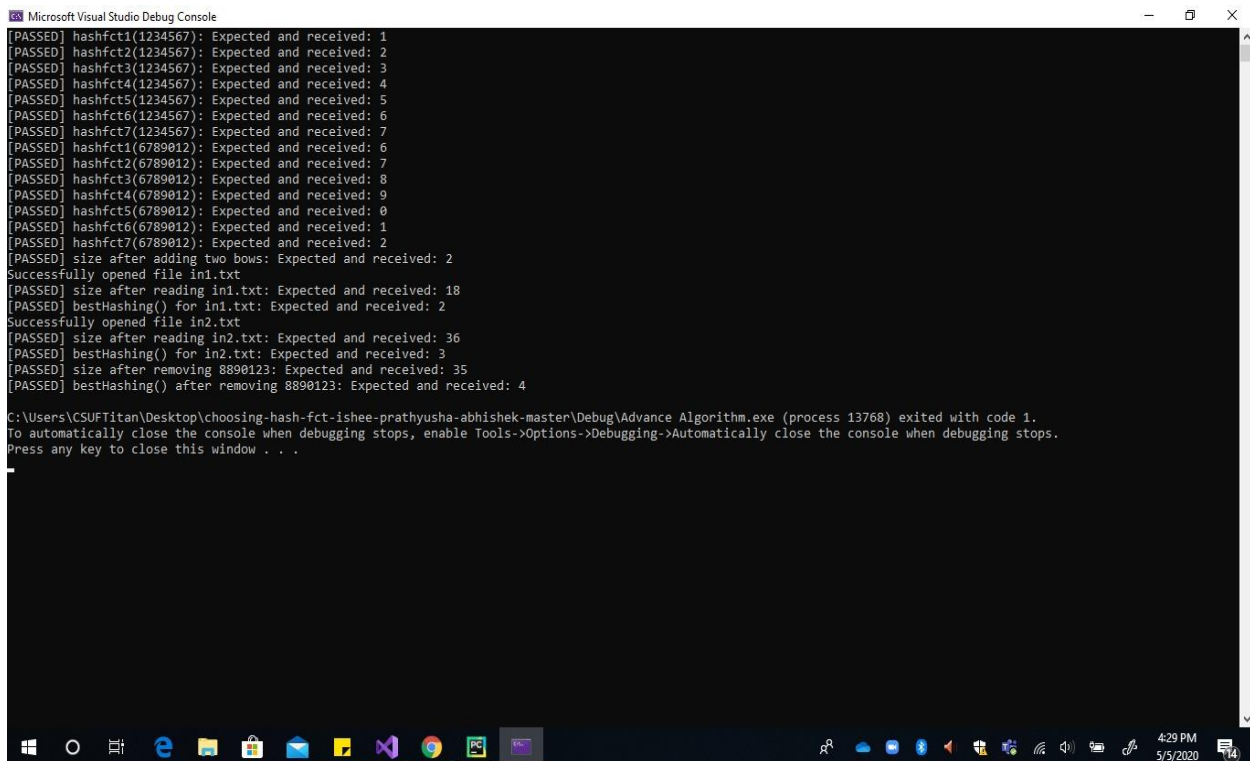
\*.cpp files (Itemcollection.cpp;main.cpp)are the input files that contain the appropriate code for reading a large number of distinct product numbers (7-digit each) and deciding which digit among the seven gives the best balanced storage of the pairs of glasses using hash functions and hash table which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



```
Microsoft Visual Studio Debug Console
[PASSED] hashfct1(1234567): Expected and received: 1
[PASSED] hashfct2(1234567): Expected and received: 2
[PASSED] hashfct3(1234567): Expected and received: 3
[PASSED] hashfct4(1234567): Expected and received: 4
[PASSED] hashfct5(1234567): Expected and received: 5
[PASSED] hashfct6(1234567): Expected and received: 6
[PASSED] hashfct7(1234567): Expected and received: 7
[PASSED] hashfct1(6789012): Expected and received: 6
[PASSED] hashfct2(6789012): Expected and received: 7
[PASSED] hashfct3(6789012): Expected and received: 8
[PASSED] hashfct4(6789012): Expected and received: 9
[PASSED] hashfct5(6789012): Expected and received: 0
[PASSED] hashfct6(6789012): Expected and received: 1
[PASSED] hashfct7(6789012): Expected and received: 2
[PASSED] size after adding two bows: Expected and received: 2
Successfully opened file in1.txt
[PASSED] size after reading in1.txt: Expected and received: 18
[PASSED] bestHashing() for in1.txt: Expected and received: 2
Successfully opened file in2.txt
[PASSED] size after reading in2.txt: Expected and received: 36
[PASSED] bestHashing() for in2.txt: Expected and received: 3
[PASSED] size after removing 8890123: Expected and received: 35
[PASSED] bestHashing() after removing 8890123: Expected and received: 4

C:\Users\CSUFTitan\Desktop\choosing-hash-fct-ishee-prathyusha-abhishek-master\Debug\Advance Algorithm.exe (process 13768) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```