Project 2: Choosing a Hash Function

Group Members:

Thomas Ngo	tngo0508@csu.fullerton.edu
Rachana Chittari	chittari.rachana@csu.fullerton.edu
Uday Margadi	uday5746@csu.fullerton.edu

Report

This project is mainly about deciding which digit leads to the most balanced hashtable within a given hash code of a pair of glasses. In the end, we will be comparing seven hash tables (which differ only in their hash function) for finding out the best hashing.

There are seven hash functions hashfct1, hashfct2, hashfct3, hashfct4, hashfct5, hashfct6, hashfct7 which takes barcode as input and returns a hash value based of digits.

For the given problem, read from the input files about one pair of glasses using readTextFile() line by line and for each line of input, create an Item object and insert them into each of the seven hash tables where it has the product number as key and Item object as the value using addItem() function. Since all the seven hash tables are initialized with seven different hash functions and bucket size, the key will be used as input to the hash function.

The function removeItem() is used to remove the pair of glasses specified by the barcode from the hash table. If the barcode is not found, it will return false otherwise, will erase the barcode from the hash table.

The function bestHashing() is used to calculate the best balance for every seven hashtables and then identifying the hashtable with the best balance should go into this method. Here, balance is defined as the difference between the sizes of the largest bucket and smallest bucket.

Pseudocode for Overall Algorithm

```
// Read the text file of input

Open file

Read itemColor, itemShape, itemBrand, barCode
```

Pseudocode for Add Item

```
// Call addItem() to add these values into the hash table.
```

```
addItem(itemColor, itemShape, itemBrand, barcode);
```

```
//Add Item adds the item object into the given hashtables.
```

Make an Item object with all input values and add them in all hash tables with barcode as key and Item object as the value.

```
struct Item current_item = {itemColor, itemShape, itemBrand, barcode};
hT1[barcode] = current_item;
hT2[barcode] = current_item;
hT3[barcode] = current_item;
hT4[barcode] = current_item;
hT5[barcode] = current_item;
hT6[barcode] = current_item;
hT7[barcode] = current_item;
```

Pseudocode for Hash Function

Hash functions takes barcode as input and return n^{th} digit of the barcode. As there are seven hash functions and length of barcode is 7, ‘n’ value ranges from 0 to 6.

```
unsigned int hashfct1(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode / 1000000;
  d %= 10;
  return d;
}
```

The above hash function takes barcode as input and returns 1st digit of barcode as hash value.

```
unsigned int hashfct2(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode / 100000;
  d %= 10;
  return d;
}
```

The above hash function takes barcode as input and returns 2nd digit of barcode as hash value.

```
unsigned int hashfct3(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode / 10000;
  d %= 10;
  return d;
}
```

The above hash function takes barcode as input and returns 3rd digit of barcode as hash value.

```
unsigned int hashfct4(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode / 1000;
  d %= 10;
  return d;
}
```

The above hash function takes barcode as input and returns 4rth digit of barcode as hash value.

```
unsigned int hashfct5(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode / 100;
  d %= 10;
  return d;
}
```

The above hash function takes barcode as input and returns 5th digit of barcode as hash value.

```
unsigned int hashfct6(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode / 10;
  d %= 10;
  return d;
}
```

The above hash function takes barcode as input and returns 6th digit of barcode as hash value.

```
unsigned int hashfct7(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode % 10;
  return d;
}
```

The above hash function takes barcode as input and returns 7th digit of barcode as hash value.

Pseudocode for Remove Item

//Use the removeItem function in order to remove the barcode from the hash tables.

```
for (unsigned int i = 0; i < 8; ++i) {
  switch (i) {
```

```
case 0:
  found = hT1.find(barcode);
  if (found == hT1.end()) return false;
  hT1.erase(barcode);
  break;

case 1:
  found = hT2.find(barcode);
  if (found == hT2.end()) return false;
  hT2.erase(barcode);
  break;

case 2:
  found = hT3.find(barcode);
  if (found == hT3.end()) return false;
  hT3.erase(barcode);
  break;

case 3:
  found = hT4.find(barcode);
  if (found == hT4.end()) return false;
  hT4.erase(barcode);
  break;

case 4:
  found = hT5.find(barcode);
  if (found == hT5.end()) return false;
  hT5.erase(barcode);
  break;

case 5:
  found = hT6.find(barcode);
  if (found == hT6.end()) return false;
  hT6.erase(barcode);
  break;
```

```
      case 6:
        found = hT7.find(barcode);
        if (found == hT7.end()) return false;
        hT7.erase(barcode);
        break;

      default:
        break;
    }
  }

  return true;
}
```

It iterates through all the hash tables and searches for the barcode. If the barcode is found in the hash table, the function will erase the barcode from the hash table.

Pseudocode for Best Hashing

```
//Use the best hashing function in order to calculate the best balance in all hashtables.

        Declare an array balance

        Initialise balance[0] = 0

        Declare three variables min_loc, max_loc, and min_val;

//Since we have seven hashtables, let’s iterate all the hashtables to find the best balance.

        for(i = 1; i < 8; ++i)

          Declare an array bucket_vals[10]
```

//Since we have 10 buckets in each hashtable, let’s iterate over them to find min_val and max_value of buckets and find the difference between them. In each table, we will find sizes of the bucket. Then find out the balance which is the difference between maximum bucket value and minimum bucket value.

```
   for(j = 0; j < 10; ++j)
     switch (i) {
                 case 1:
                     bucket_vals[j] = hT1.bucket_size(j);
           Break;
 }
```

//For each value of ‘j’, based on ‘i’ value, find out the bucket size and push it into bucket_vals[10] array.

//Once it is done, for each value of ‘i’, find out the maximum and minimum value of bucket_vals array.

```
  min_loc = *std::min_element(bucket_vals, bucket_vals + 10);
  max_loc = *std::max_element(bucket_vals, bucket_vals + 10);
```

//Push the balance value into the balance[10] array.

```
  Balance[i] =  max _ loc - min_loc
```

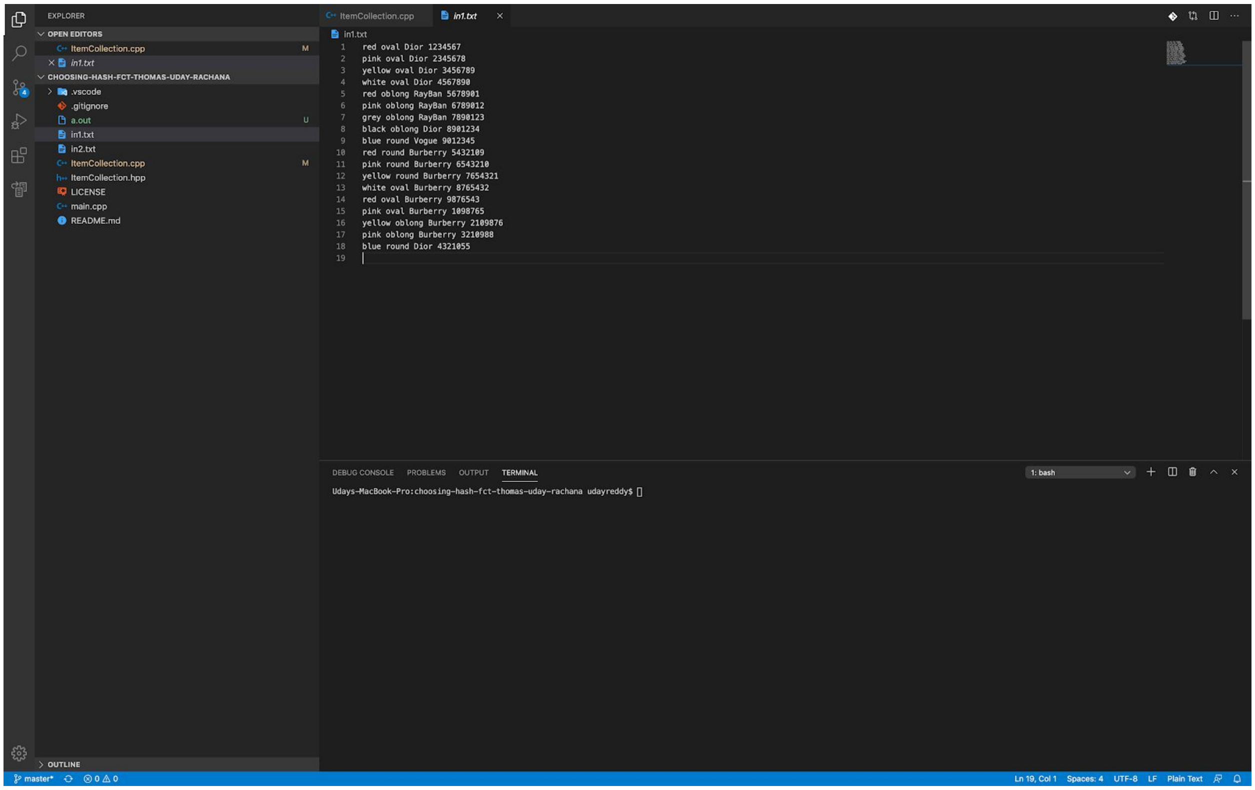
//Let’s consider the first element in the balanced array as a minimum value and compare with the rest of the array and update it’s value if it is less than iterated value.

```
min _val = balance[i]
             for(i = 2; i < 8; i++)
              If min_val > balance[i]:
                          //update minimum value
                          Min_val = balance[i]
```

```
        //return min_val
                Return min_val
```

Screenshots

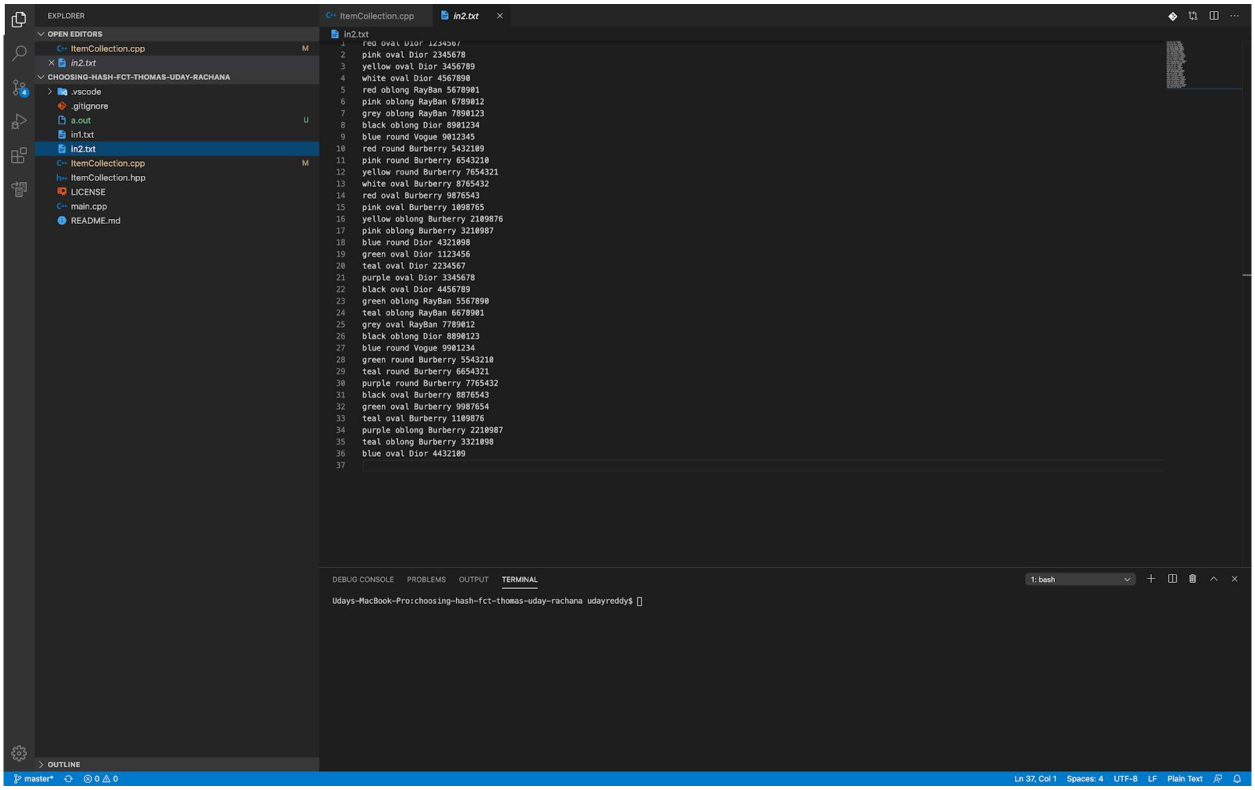
ScreenShot 1: in1.txt



```
red oval Dior 1234567
pink oval Dior 2345678
yellow oval Dior 3456789
white oval Dior 4567890
red oblong RayBan 5678901
pink oblong RayBan 6789012
grey oblong RayBan 7890123
black oblong Dior 8901234
blue round Vogue 9012345
red round Burberry 5432109
pink round Burberry 6543210
yellow round Burberry 7654321
white oval Burberry 8765432
red oval Burberry 9876543
pink oval Burberry 1098765
yellow oblong Burberry 2109876
pink oblong Burberry 3210988
blue round Dior 4321055
```

```
Udays-MacBook-Pro:choosing-hash-fct-thomas-uday-rachana udayreddy$ 
```

ScreenShot 2: in2.txt

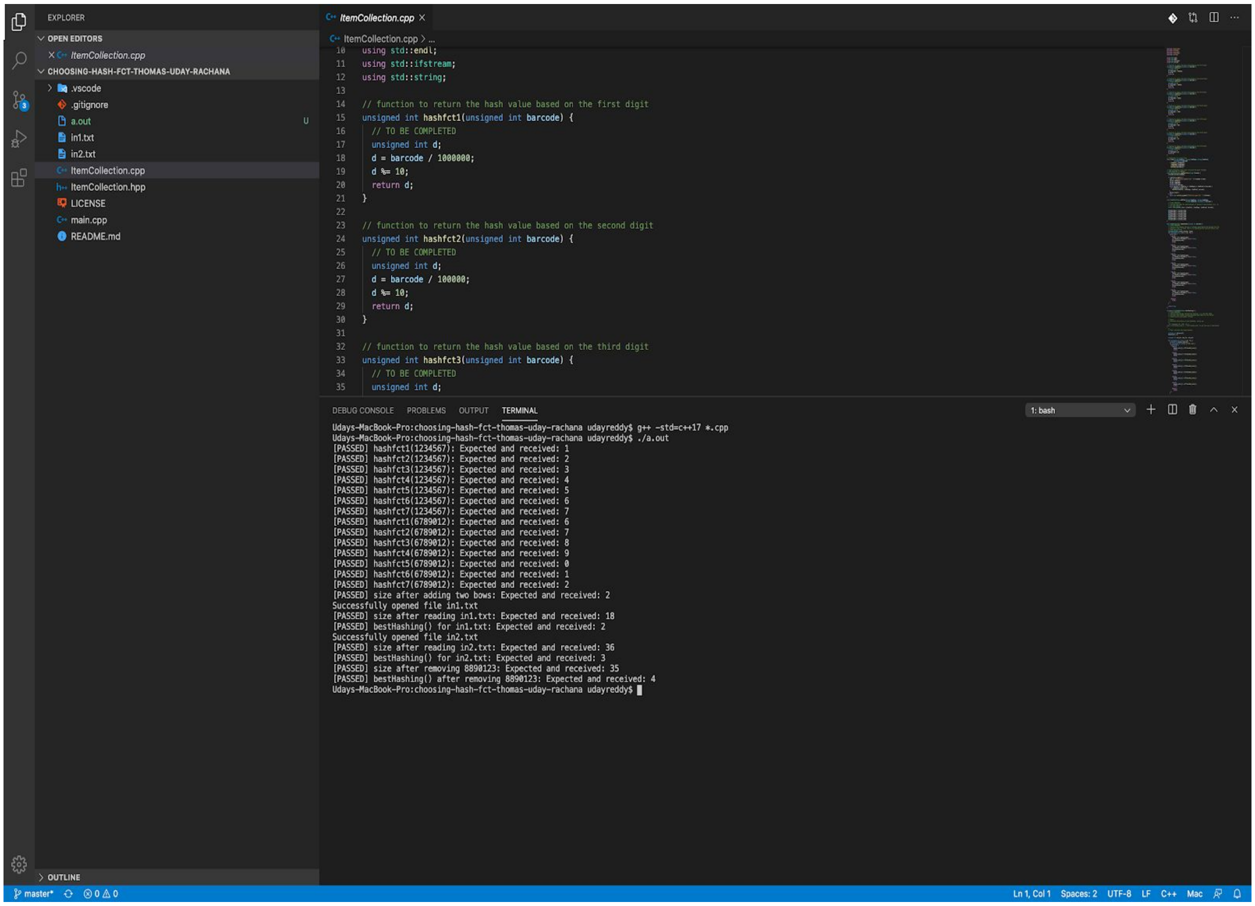


```
red oval Dior 1234567
pink oval Dior 2345678
yellow oval Dior 3456789
white oval Dior 4567890
red oblong RayBan 5678901
pink oblong RayBan 6789012
grey oblong RayBan 7890123
black oblong Dior 8901234
blue round Vogue 9012345
red round Burberry 5432109
pink round Burberry 6543210
yellow round Burberry 7654321
white oval Burberry 8765432
red oval Burberry 9876543
pink oval Burberry 1098765
yellow oblong Burberry 2109876
pink oblong Burberry 3210987
blue round Dior 4321098
green oval Dior 1123456
teal oval Dior 2234567
purple oval Dior 3345678
black oval Dior 4456789
green oblong RayBan 5567890
teal oblong RayBan 6678901
grey oval RayBan 7789012
black oblong Dior 8890123
blue round Vogue 9901234
green round Burberry 5543210
teal round Burberry 6654321
purple round Burberry 7765432
black oval Burberry 8876543
green oval Burberry 9987654
teal oval Burberry 1109876
purple oblong Burberry 2210987
teal oblong Burberry 3321098
blue oval Dior 4432109

```

```
Udays-MacBook-Pro:choosing-hash-fct-thomas-uday-rachana udayreddy$ 
```

ScreenShot 3: Output



```cpp
using std::endl;
using std::ifstream;
using std::string;

// function to return the hash value based on the first digit
unsigned int hashfct1(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode / 1000000;
  d %= 10;
  return d;
}

// function to return the hash value based on the second digit
unsigned int hashfct2(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
  d = barcode / 100000;
  d %= 10;
  return d;
}

// function to return the hash value based on the third digit
unsigned int hashfct3(unsigned int barcode) {
  // TO BE COMPLETED
  unsigned int d;
```

```
Udays-MacBook-Pro:choosing-hash-fct-thomas-uday-rachana udayreddy$ g++ -std=c++17 *.cpp
Udays-MacBook-Pro:choosing-hash-fct-thomas-uday-rachana udayreddy$ ./a.out
[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
Udays-MacBook-Pro:choosing-hash-fct-thomas-uday-rachana udayreddy$ 
```