Group members:

Anant Jain jain.anant@csu.fullerton.edu

Harsh Bodgal harsh481997@csu.fullerton.edu

Hemil Prajapati hemilhp22@csu.fullerton.edu

*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-fct6 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 6 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.

How to run the code:

The *.cpp files (SensorCluster.cpp;main.cpp) are the input files that contain the appropriate code for reading a large number of different product numbers (6-digit each) and determining which digit among the six gives the best balanced storage of the pairs of glasses using hash functions and hash tables, which can be executed using Visual Studio with the built-in option "Run" from the menu bar. Otherwise, on the command prompt, type "gcc" (the C-compiler) followed by the complete name of your program. Your source code will be compiled and an executable file will be created on your desktop.

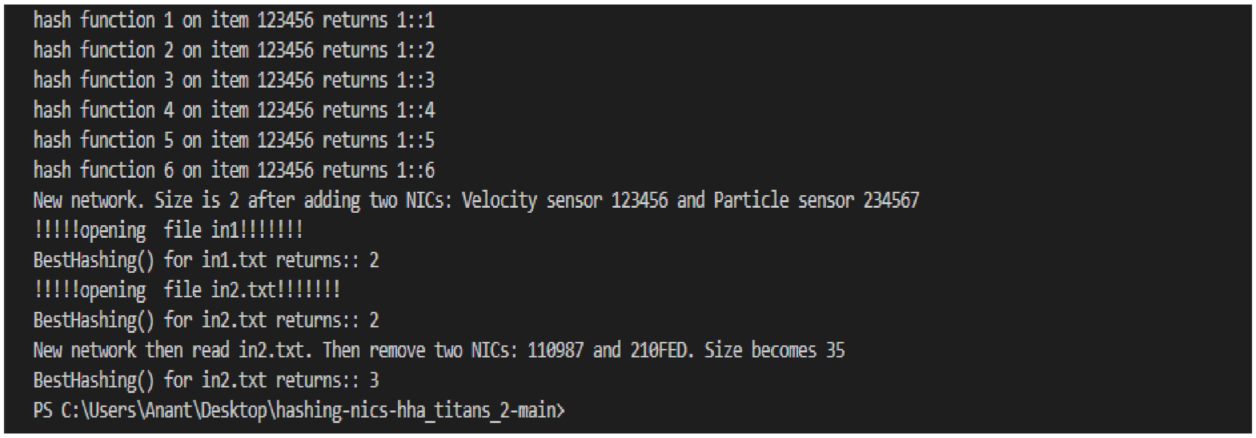
Steps:

1. The user must supply specific inputs for each of the Cpp files mentioned above.

2. The code would then be processed to determine the best-balanced storage after reading the different product numbers.

3. After reading/removing relevant storage pairs, the result would show the best hashing function.

Snapshot of Code: -



```
hash function 1 on item 123456 returns 1::1
hash function 2 on item 123456 returns 1::2
hash function 3 on item 123456 returns 1::3
hash function 4 on item 123456 returns 1::4
hash function 5 on item 123456 returns 1::5
hash function 6 on item 123456 returns 1::6
New network. Size is 2 after adding two NICs: Velocity sensor 123456 and Particle sensor 234567
!!!!!opening  file in1!!!!!!!
BestHashing() for in1.txt returns:: 2
!!!!!opening  file in2.txt!!!!!!!
BestHashing() for in2.txt returns:: 2
New network then read in2.txt. Then remove two NICs: 110987 and 210FED. Size becomes 35
BestHashing() for in2.txt returns:: 3
PS C:\Users\Anant\Desktop\hashing-nics-hha_titans_2-main>
```