```
/*
	Bee Cha
	CSCI 117 Lab4

	This lab is to exercise matching duplicate keys in a data file
	When a duplicate key is found, we push the label corresponding
	to the key into a vector. The data file is substantially large
	so I will be using a "testfile.txt" for testing short and small
	data. The output of the result should have all the matching keys
	with the corresponding labels.

	Label1, Label2, Label3, ..... LabelN
	"KEY"

	So if I had a data file:

	Label1
	ABCD
	Label2
	ABCE
	Label3
	ABCF
	Label4
	ABCD
	Label5
	ABCG

	There is two duplicate keys of ABCD from Label1 and Label4, then the output should be:

	Label1, Label4
	ABCD
	Label2
	ABCE
	Label3
	ABCF
	Label5
	ABCG

	Note that Label4 and its key should be deleted and not reprinted again from the output
	*I'm having trouble with erasing from the unordered map*
*/

#include <iostream>
#include <unordered_map>
#include <string>
#include <vector>
#include <fstream>

using namespace std;

// Load data file into the input stream
ifstream input_file("testfile.txt");
//ifstream input_file("Prog4-data");

//------Unordered_map Functions------//
// find, count, size, insert, erase
// Unordered_map is to search for duplicated keys
                                          //Labels  , Keys
typedef unordered_map< string , string  > m;
```

```
//------Vector Functions------//
// push_back, pop_back, insert, erase, size, front, back
// Vector is to push all the labels of duplicated keys
vector<string> labels;
vector<string> keys;


int main()
{

       m test;
       string s_label, s_key;


       // This while loop will populate all the labels and keys into the unordered map list
       // It will first check the stream for a label, if there's a label then store it to 
s_label
       // Then it will check the stream for a key, if there's a key then store it to s_key
       // So that will then be inserted into the unordered map by calling the insert method
       // If no label or key is found in the input stream, then exit the loop
       while ( (getline(input_file, s_label) && (getline(input_file, s_key)) ) )
       {
              test.insert({ s_label, s_key });
       }

       // Now that the labels and keys are loaded into the unordered map, we need to check for
       // duplicate keys. First, push the first label into the label vector and copy the key
       // to a string variable, then use that variable to iterate through the unordered map to 
find
       // instances of the duplicate key. Each time it comes across the duplicate key, push the 
label
       // of that duplicate key into the label vector, but don't forget to concatenate the 
label with
       // a comma before hand so that the output can look like (Label1, Label2, Label3, ...)
       // When the unordered map iterates to the end of the map, output the results
       // --
       // The result: Cout the vector label ( v.back() ) in a loop and keep popping it until 
it's empty
       // Then output the key associated with the label(s). cout map->second will grab the 
second
       // element of the unordered map.


       for (auto& it : test)
       {
              s_label = it.first;
              s_key = it.second;
              keys.push_back(s_key);
              //labels.push_back(s_label);
              //test.erase(it.first); gives me runtime error
              // My guess is when deleting a tuple from the unordered map,
              // the map is rehashing and causing me error

              for (auto& iit : test)
              {
                     if (iit.second == s_key)
                     {
                            labels.push_back(iit.first);
                     }
```

```
		}

		// Output the label vector until it's empty
		// Output the associated key with the labels
		while (labels.size() != 0)
		{
			cout << labels.back();
			labels.pop_back();
			if (labels.size() >= 1)
			{
				cout << ", ";
			}

		}
		// Now for the key
		cout << endl << s_key << endl;

	}


	// ----- Just some notes for iterating an unordered map ----- //
	/*
	// This for function is the iteration for the unordered map
	// x is the iterator, x.first is the label, x.second is the key
	for (auto& x : test)
	{
		s_label = x.first;
		s_key = x.second;

		//Push the label into the label vector then find the duplicate keys
		labels.push_back(s_label);
		keys.push_back(s_key);

		cout << s_label << endl << s_key << endl;

	}
	*/


	return 0;
}
```

Output Examples using my own data file

```
Label4, Label1
ABCD
Label2
ABCE
Label3
ABCF
Label4, Label1
ABCD
Label5
ABCG

Label3, Label1
1111
Label2
0000
Label3, Label1
1111
Label5, Label4
1010
Label5, Label4
1010
```