```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

ifstream fin;
struct place {
	int code = 0;
	string state = "";
	string name = "";
	int pop = 0; //population
	double area = 0;
	double lat = 0; //latitude
	double lon = 0; //longitude
	int inter = 0; //intersection code
	double dist = 0; //distance to intersection
};
struct ht_items {
	string key; //key will be place name
	place data; //data will be all the data associated to that name
	ht_items(place p) {
		key = p.name;
		data = p;
	}
};
class LL {
protected:
	struct Link {
		ht_items* items;
		Link* next;
		Link(ht_items* i, Link* n) {
			items = i;
			next = n;
		}
	};
	Link* first, * last;
	int size;
public:
	LL() {
		first = NULL;
		last = NULL;
		size = 0;
	}
	void add_to_front(ht_items* x) {
		first = new Link(x, first);
		if (last == NULL) {
			last = first;
		}
		size++;
	}
	void add_to_back(ht_items* x) {
		if (last != NULL) {
			last->next = new Link(x, NULL);
			last = last->next;
		}
```

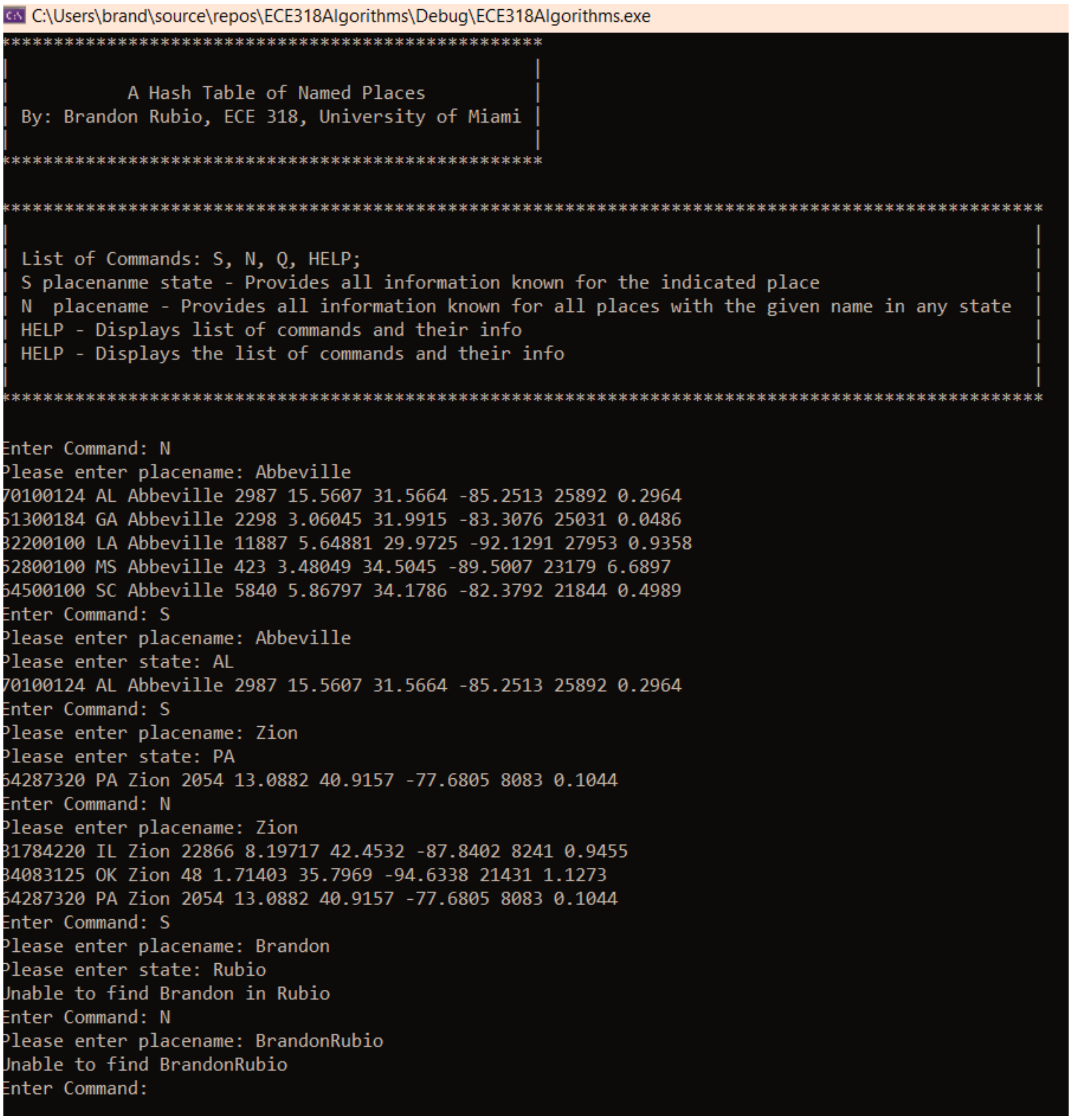
```
			else {
				last = new Link(x, last);
				first = last;
			}
			size++;
		}
		place find(string name, string state) { //retuns the named place
			place p;
			place error;
			Link* ptr = first;
			bool found = false;
			while (ptr != NULL) {
				if ((ptr->items->data.name == name) && (ptr->items->data.state == 
state)) {
					p = ptr->items->data;
					found = true;
				}
				ptr = ptr->next;
			}
			if (found == false) {
				error.code = -1;
				return error;
			}
			return p;
		}
		place find(string name) { //retuns info of all same named place
			place p;
			place error;
			Link* ptr = first;
			bool found = false;
			while (ptr != NULL) {
				if (ptr->items->data.name == name) {
					p = ptr->items->data;
					cout << p.code << " " << p.state << " " << p.name << " " << 
p.pop << " " << p.area << " " << p.lat << " " << p.lon << " " << p.inter << " " << p.dist 
<< endl;
					found = true;
				}
				ptr = ptr->next;
			}
			if (found == false) {
				error.code = -1;
				return error;
			}
			return p;
		}
};
class HT {//Hash table
private:
		LL item_list[30000]; //anything higher exceeds stack size of compiler (Visual 
Studio)
		int size;
		int count;
		int hash_function(string key) {
			const int p = 53;
			const int m = 30000;
			long hash_value = 0;
			long p_pow = 1;
			for (char c : key) {
				hash_value = (hash_value + (c - '\'' + 1) * p_pow) % m;
				p_pow = (p_pow * p) % m;
			}
			//cout << "key: " << key << " hash: " << hash_value << endl;
			return hash_value;
		}
public:
		HT() {
			size = 30000;
			count = 0;
		}
		void add(place p);
		void readFile(string f);
		void find(string name, string state);
		void find(string name);
};
void HT::add(place p) {
		ht_items* pToItems = new ht_items(p);
		int position = hash_function(pToItems->data.name);
		item_list[position].add_to_back(pToItems);
		count++;
}
void HT::readFile(string f) {

		fin.open(f);
		if (fin.fail()) {
			cout << "Unable to open File" << endl;
		}
		int j = 0;
		while (!fin.fail()) {
			place p;
			string s, numbers = "", alphaSD = ""; // alphaSD = alphabetical characters,
spaces, and dashes accepted

			getline(fin, s);
			if (s == "") {
				break;
			}
			for (int i = 0; i < s.length(); i++) {
				if (!(s[i] >= 'A' && s[i] <= 'Z') && !(s[i] >= 'a' && s[i] <= 'z'))
{//anything except alphabetical characters

					if ((i < (s.length() - 1)) && i < 76) {// double space filter
for the first half (allows for an easier time separating each component later)
						if (!((s[i] == ' ') && (s[i + 1] == ' '))) {
							numbers = numbers + s[i];
						}
					}
					else {
						numbers = numbers + s[i];
					}
				}

				if (!(s[i] >= '0' && s[i] <= '9')) {//takes anything except numbers

					if (i < (s.length() - 1)) {//double space filter
						if (!((s[i] == ' ') && (s[i + 1] == ' '))) {
```

```
                                        alphaSD = alphaSD + s[i];
                                }
                        }
                        else {
                                alphaSD = alphaSD + s[i];
                        }
                }

        }

        string code = "";
        string state = "";
        string name = "";
        string pop = ""; //population
        string area = "";
        string lat = ""; //latitude
        string lon = ""; //longitude
        string inter = ""; //intersection code
        string dist = ""; //distance to intersection

        //cout << "numbers: " << numbers << endl;
        //cout << "alphaSD: " << alphaSD << endl;
        for (int i = 0; i < alphaSD.length(); i++) {
                if (i <= 1) {
                        state = state + alphaSD[i];
                }
                if ((i < (alphaSD.length() - 1)) && (i > 1)) {//double space filter
                        if (((alphaSD[i] == ' ') && (alphaSD[i + 1] == ' '))) {
                                break;
                        }
                        else {
                                name = name + alphaSD[i];
                        }
                }
        }
        //cout << "name: " << name << endl;
        int k = 0;
        while (numbers[k] != ' ') {
                code = code + numbers[k];
                k++;
        }
        //cout << "code: " << code << endl;
        while (numbers[k] == ' ' || numbers[k] == '-' || numbers[k] == '.' ||
numbers[k] == '\'') {
                k++;
        }
        while (numbers[k] != ' ') {
                pop = pop + numbers[k];
                k++;
        }
        //cout << "pop: " << pop << endl;
        while (numbers[k] == ' ') {
                k++;
        }
        while (numbers[k] != ' ') {
                area = area + numbers[k];
                k++;
        }
```

```
//cout << "area: " << area << endl;
while (numbers[k] == ' ') {
       k++;
}
while ((numbers[k] != '-') && (numbers[k] != ' ')) {
       lat = lat + numbers[k];
       k++;
}
//cout << "lat: " << lat << endl;
while (numbers[k] == '-') {
       lon = lon + numbers[k];
       k++;
}
while (numbers[k] == ' ') {
       k++;
}
while (numbers[k] != '.') {
       lon = lon + numbers[k];
       k++;
}
lon = lon + numbers[k]; //adds the '.'
k++;
int n = k;
for (n; n < k + 6; n++) {
       lon = lon + numbers[n];
}
//cout << "lon: " << lon << endl;
k = k + 6;
n = k;
for (n; n < k + 5; n++) {
       if ((numbers[n] != ' ')) {
              inter = inter + numbers[n];
       }
}
//cout << "inter: " << inter << endl;
k = k + 5;
n = k;
for (n; n < numbers.length(); n++) {
       if ((numbers[n] != ' ')) {
              dist = dist + numbers[n];
       }
}
//cout << "inter: " << inter << endl;

p.code = stoi(code);
p.state = state;
p.name = name;
p.pop = stoi(pop);
p.area = stod(area);
p.lat = stod(lat);
p.lon = stod(lon);
p.inter = stoi(inter);
p.dist = stod(dist);


//cout << "p.state: " << p.state << endl;
//cout << "p.name: " << p.name << endl;
//cout << "p.code: " << p.code << endl;
```

```
              //cout << "p.pop: " << p.pop << endl;
              //cout << "p.area: " << p.area << endl;
              //cout << "p.lat: " << p.lat << endl;
              //cout << "p.long: " << p.lon << endl;
              //cout << "p.inter: " << p.inter << endl;
              //cout << "p.dist: " << p.dist << endl;

              add(p);
       }

       fin.close();

}
void HT::find(string name, string state) {
       place p;
       int position = hash_function(name);
       p = item_list[position].find(name, state);
       if (p.code == -1) {
              cout << "Unable to find " << name << " in " << state << endl;
       }
       else {
              cout << p.code << " " << p.state << " " << p.name << " " << p.pop << " " <<
p.area << " " << p.lat << " " << p.lon << " " << p.inter << " " << p.dist << endl;
       }

}
void HT::find(string name) {
       place p;
       int position = hash_function(name);
       p = item_list[position].find(name);
       if (p.code == -1) {
              cout << "Unable to find " << name << endl;
       }
}
void titlePage() {
       cout << "***************************************************" << endl;
       cout << "|                                                 |" << endl;
       cout << "|            A Hash Table of Named Places         |" << endl;
       cout << "| By: Brandon Rubio, ECE 318, University of Miami |" << endl;
       cout << "|                                                 |" << endl;
       cout << "***************************************************" << endl;
       cout << endl;
}
void LOC() {//list of commands
       cout << "List of Commands: S, N, Q" << endl;
}
void LOCI() {//list of commands and their info
       cout <<
"*****************************************************************************************
**********" << endl;
       cout << "|
|" << endl;
       cout << "| List of Commands: S, N, Q, HELP;
|" << endl;
       cout << "| S placenanme state - Provides all information known for the indicated
place                    |" << endl;
       cout << "| N  placename - Provides all information known for all places with the
given name in any state  |" << endl;
```

```
       cout << "| HELP - Displays list of commands and their info
|" << endl;
       cout << "| HELP - Displays the list of commands and their info
|" << endl;
       cout << "|
|" << endl;
       cout <<
"*****************************************************************************************
**********" << endl;
       cout << endl;
}
int main() {
       HT HashTable;
       HashTable.readFile("/home/www/class/een318/named-places.txt");
       string input;
       titlePage();
       LOCI();
       while (true) {
              cout << "Enter Command: ";
              cin >> input;
              if (input == "S") {
                     string placename, state;
                     cout << "Please enter placename: ";
                     cin >> placename;
                     cout << "Please enter state: ";
                     cin >> state;
                     HashTable.find(placename, state);
              }
              else if (input == "N") {
                     string placename;
                     cout << "Please enter placename: ";
                     cin >> placename;
                     HashTable.find(placename);
              }
              else if (input == "Q") {
                     cout << "Exiting program" << endl;
                     exit(1);
              }
              else if (input == "HELP") {
                     LOCI();
              }
              else {
                     cout << "Invalid input, please type \"HELP\" for a list of commands"
<< endl;
              }
       }
return 0;
}
```



```
C:\Users\brand\source\repos\ECE318Algorithms\Debug\ECE318Algorithms.exe
**************************************************
|                                                |
|          A Hash Table of Named Places          |
| By: Brandon Rubio, ECE 318, University of Miami |
|                                                |
**************************************************

**************************************************************************************************
|                                                                                                |
| List of Commands: S, N, Q, HELP;                                                               |
| S placenanme state - Provides all information known for the indicated place                   |
| N  placename - Provides all information known for all places with the given name in any state |
| HELP - Displays list of commands and their info                                                |
| HELP - Displays the list of commands and their info                                            |
|                                                                                                |
**************************************************************************************************

nter Command: N
lease enter placename: Abbeville
0100124 AL Abbeville 2987 15.5607 31.5664 -85.2513 25892 0.2964
1300184 GA Abbeville 2298 3.06045 31.9915 -83.3076 25031 0.0486
2200100 LA Abbeville 11887 5.64881 29.9725 -92.1291 27953 0.9358
2800100 MS Abbeville 423 3.48049 34.5045 -89.5007 23179 6.6897
4500100 SC Abbeville 5840 5.86797 34.1786 -82.3792 21844 0.4989
nter Command: S
lease enter placename: Abbeville
lease enter state: AL
0100124 AL Abbeville 2987 15.5607 31.5664 -85.2513 25892 0.2964
nter Command: S
lease enter placename: Zion
lease enter state: PA
4287320 PA Zion 2054 13.0882 40.9157 -77.6805 8083 0.1044
nter Command: N
lease enter placename: Zion
1784220 IL Zion 22866 8.19717 42.4532 -87.8402 8241 0.9455
4083125 OK Zion 48 1.71403 35.7969 -94.6338 21431 1.1273
4287320 PA Zion 2054 13.0882 40.9157 -77.6805 8083 0.1044
nter Command: S
lease enter placename: Brandon
lease enter state: Rubio
nable to find Brandon in Rubio
nter Command: N
lease enter placename: BrandonRubio
nable to find BrandonRubio
nter Command:
```