```
*********************************************************
*   PROGRAMMED BY : Carlos Aguilera
*   CLASS         : CS1B
*   SECTION       : MW: 7:30p - 9:50p
*   LAB #8        : Lab Linked Lists
*********************************************************

Enter name of student to search: John Smith

FOUND
--------
Name:   John Smith
Age:    20
Major:  Math
GPA:    3.5

John Smith
------------------
20
Math
3.5

Anna White
------------------
19
English
3.2

Paul Johnson
------------------
18
Physics
3.7

Connor Darling
------------------
18
Computer Science
4

Carlos Aguilera
------------------
21
Computer Science
4.5

Rand Om
------------------
33
Math
2.3

Coo Lguy
------------------
10
Science
3.4

Larry Chad
```

```
------------------
14
Science
2.3

POPPING
Name:   John Smith
Age:    20
Major:  Math
GPA:    3.5

Average GPA: 3.34286

Anna White
------------------
19
English
3.2

Paul Johnson
------------------
18
Physics
3.7

Connor Darling
------------------
18
Computer Science
4

Carlos Aguilera
------------------
21
Computer Science
4.5

Rand Om
------------------
33
Math
2.3

Coo Lguy
------------------
10
Science
3.4

Larry Chad
------------------
14
Science
2.3
```

```cpp
#ifndef _HEADER_H_
#define _HEADER_H_
#include <iostream>
#include <string>
#include <fstream>
#include <iomanip>
#include <map>
#include <unordered_map>
#include "studentNode.h"

using namespace std;

void readData(StudentNode** head, map<string, StudentNode> &studentData);
void heading();
void dispList(StudentNode* &head);
void pop(StudentNode** head, map<string, StudentNode> &studentData);
void search(StudentNode* &head, map<string, StudentNode> &studentData);
void average(StudentNode** head);

#endif

#ifndef _STUDENTNODE_H_
#define _STUDENTNODE_H_
#include "header.h"

struct StudentNode {
    string name;
    int age;
    string major;
    float gpa;
    StudentNode* nextNode;
};

#endif

#include "../include/header.h"
#include "../include/studentNode.h"

int main()
{
    heading();
    StudentNode* head = nullptr;
    /*
        Using map to get myself understanding hashing and Big O(1) with hashing
        so I used an ordered map just to keep the elements in the map the same as what
        was read in the file, also made the name the id to hash
    */
    map<string, StudentNode> studentData;
    readData(&head, studentData);//a ptr to ptr for accesses the ptr in main

    search(head, studentData);
    pop(&head, studentData);
    average(&head);
    dispList(head);

    return 0;
}
```

```
58
59 #include "../include/header.h"
60 #include "../include/studentNode.h"
61
62 void readData(StudentNode** head, map<string, StudentNode> &studentData)
63 {
64     StudentNode* node = nullptr;
65     node = new StudentNode;
66     *head = node;
67     string temp;
68
69     ifstream inFile;
70     inFile.open("inFile.txt");
71     if (inFile.is_open())
72     {
73         while (!inFile.eof())
74         {
75             getline(inFile, node->name);
76             inFile >> node->age;
77             inFile.ignore(1000, '\n');
78             getline(inFile, node->major);
79             inFile >> node->gpa;
80             inFile.ignore(1000, '\n');
81             getline(inFile, temp);
82
83             if (node->name[node->name.size() - 1] == ' ')
84             {
85                 node->name = node->name.substr(0, node->name.size() - 1);
86             }
87
88             if (node->major[node->major.size() - 1] == ' ')
89             {
90                 node->major = node->major.substr(0, node->major.size() - 1);
91             }
92
93             studentData[node->name] = *node;
94
95             if (!inFile.eof())
96             {
97                 node->nextNode = new StudentNode;
98                 node = node->nextNode;
99             }else
100                node->nextNode = nullptr;
101        }
102    }else {
103        cout << "File did not open successfully!\n";
104        delete node;
105    }
106
107    inFile.close();
108 }
109
110 #include "../include/header.h"
111 #include "../include/studentNode.h"
112
113 void dispList(StudentNode* &head)
114 {
115     StudentNode* node = head;
116     while (node != nullptr)
117     {
```

```
        cout << node->name << "\n";
        cout << "------------------\n";
        cout << node->age << "\n";
        cout << node->major << "\n";
        cout << node->gpa << "\n\n";
        node = node->nextNode;
    }

}

#include "../include/header.h"
#include "../include/studentNode.h"

void pop(StudentNode** head, map<string, StudentNode> &studentData)
{
    StudentNode* node = *head;
    if (node != nullptr)
    {
        cout << left;
        cout << "POPPING\n";
        cout << setw(8) << "Name:" << node->name << "\n";
        cout << setw(8) << "Age:" << node->age << "\n";
        cout << setw(8) << "Major:" << node->major << "\n";
        cout << setw(8) << "GPA:" << node->gpa << "\n\n";
        cout << right;

        studentData.erase(node->name);
        *head = node->nextNode;
        delete node;
    }else
        cout << "The stack is empty!\n\n";

}

#include "../include/header.h"
#include "../include/studentNode.h"

void search(StudentNode* &head, map<string, StudentNode> &studentData)
{
    string key;

    cout << "Enter name of student to search: ";
    getline(cin, key);


    if (studentData.find(key) != studentData.end()) //Using .find method in
the map to find key
    {
        StudentNode node = studentData[key];

        cout << left;
        cout << "\nFOUND\n--------";
        cout << setw(9) << "\nName:" << node.name << "\n";
        cout << setw(8) << "Age:" << node.age << "\n";
        cout << setw(8) << "Major:" << node.major << "\n";
        cout << setw(8) << "GPA:" << node.gpa << "\n\n";
        cout << right;
    }else
        cout << "Student was not found!\n";
}
```

```
177
178 #include "../include/header.h"
179 #include "../include/studentNode.h"
180
181 void average(StudentNode** head)
182 {
183     StudentNode* node = nullptr;
184     node = *head;
185     int size = 0;
186     float totalGPA = 0;
187
188     while (node != nullptr)
189     {
190         totalGPA += node->gpa;
191         node = node->nextNode;
192         ++size;
193     }
194     cout << "Average GPA: " << totalGPA/size << "\n";
195 }
```