csc 326 lab 7

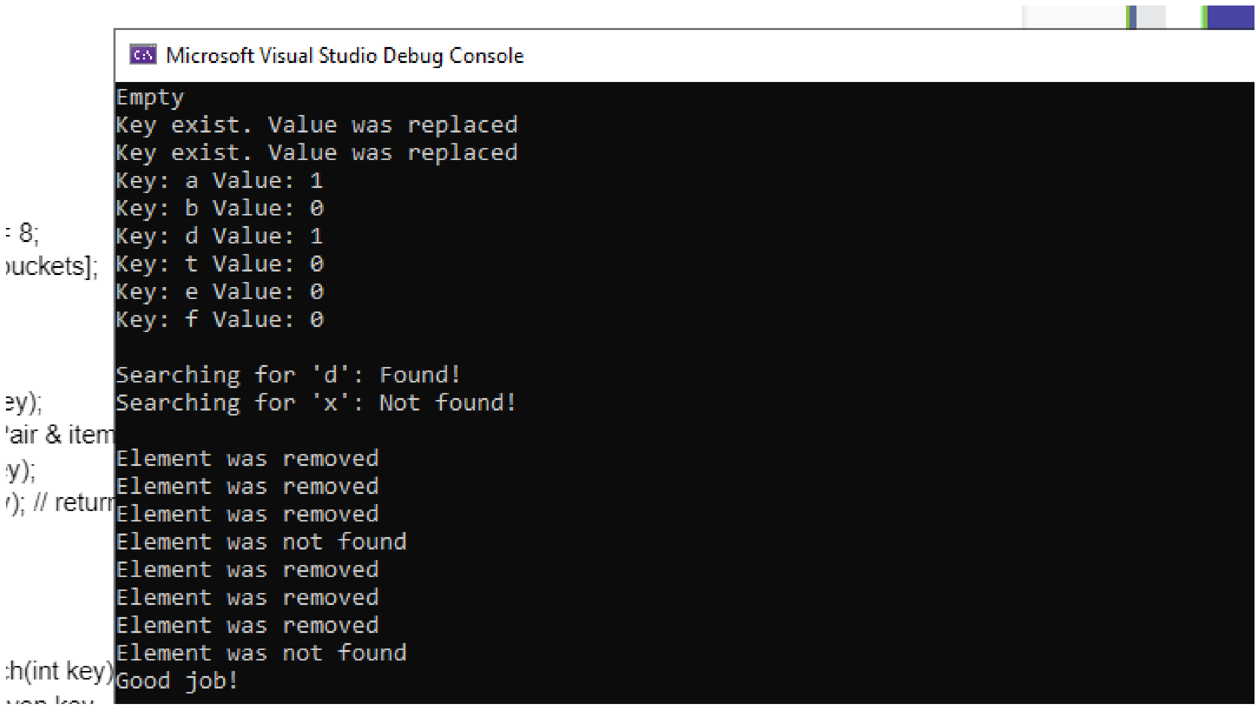
```
#include<iostream>
#include<list>
using namespace std;

struct Pair {
  char key;
  bool value;
};
class Hashmap {

  private:
    static
  const int buckets = 8;
  list < Pair > table[buckets];

  public:
    bool isEmpty();
  int hashFunc(int key);
  void insert(const Pair & item);
  void remove(int key);
  Pair search(int key); // returns pair
  void print();
};


Pair Hashmap::search(int key) {
//looking for hash of given key
    int hash = hashFunc(key);
//having the reference list at index = hash
  auto & chain = table[hash];
  for (auto it = chain.begin(); it != chain.end(); it++) {
//if key value matches given key, returning current pair
    if (it -> key == key) {
      return *it;
    }
  }
  //if the program reaches this point, then value is not found.
  Pair empty;
  //setting key to '\0' (null terminator) to indicate that the value is not found
  empty.key='\0';
  //returning it
    return empty;
```

```
}

bool Hashmap::isEmpty() {
    for (int i = 0; i < buckets; i++) {
        if (table[i].size() > 0)
            return false;
    }
    return true;
}

int Hashmap::hashFunc(int key) {
    return key % buckets;
}

void Hashmap::insert(const Pair & item) {
    int hash = hashFunc(item.key);
    auto & chain = table[hash]; // pointer to a linked list
    // pointer to a head

    bool exist = false;
    for (auto it = chain.begin(); it != chain.end(); it++) {
        if (it -> key == item.key) {
            exist = true;
            it -> value = item.value;
            cout << "Key exist. Value was replaced" << endl;
            break;
        }
    }
    if (!exist) {
        chain.emplace_back(item);
    }
}

void Hashmap::remove(int key) {
    int hash = hashFunc(key);
    auto & chain = table[hash]; // pointer to a linked list
    auto it = chain.begin();
    bool exist = false;
    for (; it != chain.end(); it++) {
        if (it -> key == key) {
            exist = true;
            it = chain.erase(it); // return a pointer to next value
            cout << "Element was removed" << endl;
            break;
        }
    }
    if (!exist) {
        cout << "Element was not found" << endl;
    }
}

void Hashmap::print() {
    for (int i = 0; i < buckets; i++) {
        if (table[i].size() == 0) continue;
        auto it = table[i].begin();
        for (; it != table[i].end(); it++)
            cout << "Key: " << it -> key <<
            " Value: " << it -> value << endl;
    }
}

int main() {
    Hashmap map;
    if (map.isEmpty()) {
        cout << "Empty" << endl;
    } else {
        cout << "Problem" << endl;
    }
    map.insert({ 'a', 0 });
        map.insert({ 'd', 0 });
        map.insert({ 'f', 0 });
        map.insert({ 'e', 0 });
        map.insert({ 'a', 1 });
        map.insert({ 't', 0 });
        map.insert({ 'd', 1 });
        map.insert({ 'b', 0 });

    map.print();

    // testing search method
    cout<<endl;
    cout<<"Searching for 'd': ";
    Pair temp = map.search('d');
    if(temp.key=='d'){
        cout<<"Found!\n";
        }else{
                cout<<"Not found!\n";
        }
        cout<<"Searching for 'x': ";
    temp = map.search('x');
    if(temp.key=='x'){
        cout<<"Found!\n";
        }else{
                cout<<"Not found!\n";
}
    cout<<endl;

    map.remove('a');
    map.remove('d');
    map.remove('f');
    map.remove('f');
    map.remove('e');
    map.remove('b');
    map.remove('t');
    map.remove('d');

    if (map.isEmpty()) {
    cout << "Good job!" << endl;
    } else {
    cout << "Problem!!!" << endl;
    }


    return 0;
}
```

Hashmap

```
class Solution {
public:
   vector<int> twoSum(vector<int>& nums, int target) {
       vector<int>ans;
       unordered_map<int,int>m;

       for(int i=0;i<nums.size();i++)
       {
           int val = target-nums[i];
           if(m.find(val)!=m.end()) // if second element is found
           {
               ans.push_back(m.find(val)->second);
               ans.push_back(i);
```

```
                break;
            }
            m.insert(pair<int,int>(nums[i],i)); // if the above criteria is not satisfied I will keep inserting
the element in the hashmap
        }
        return ans;
    }
};
```



Microsoft Visual Studio Debug Console

```
Empty
Key exist. Value was replaced
Key exist. Value was replaced
Key: a Value: 1
Key: b Value: 0
Key: d Value: 1
Key: t Value: 0
Key: e Value: 0
Key: f Value: 0

Searching for 'd': Found!
Searching for 'x': Not found!

Element was removed
Element was removed
Element was removed
Element was not found
Element was removed
Element was removed
Element was removed
Element was not found
Good job!
```

```
= 8;
uckets];
ey);
air & item
y);
); // return
h(int key)
```