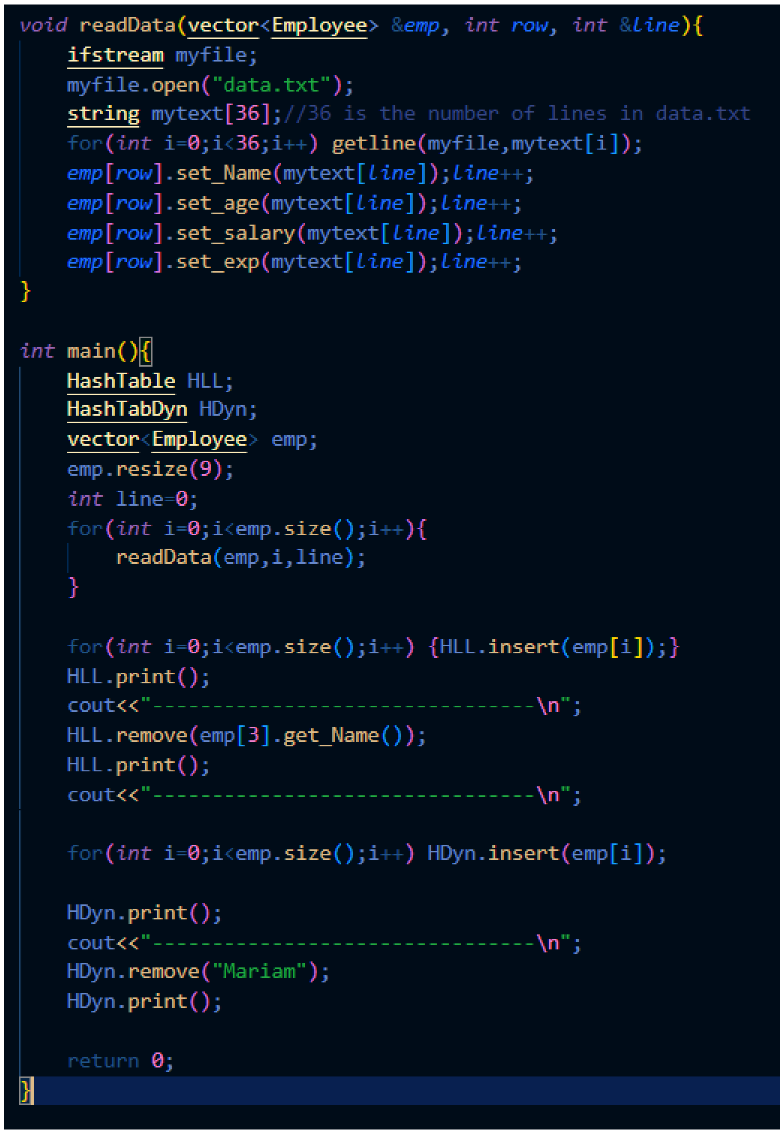
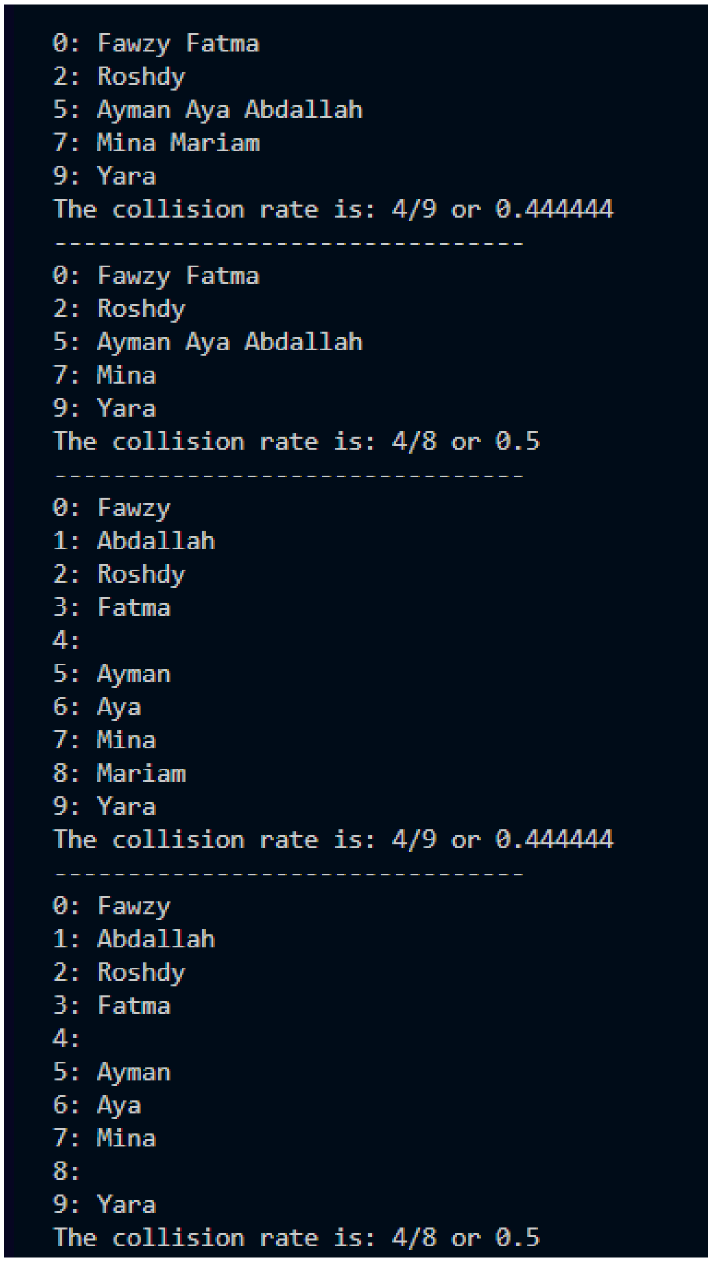
Code and Its algorithm:



```cpp
void readData(vector<Employee> &emp, int row, int &line){
    ifstream myfile;
    myfile.open("data.txt");
    string mytext[36];//36 is the number of lines in data.txt
    for(int i=0;i<36;i++) getline(myfile,mytext[i]);
    emp[row].set_Name(mytext[line]);line++;
    emp[row].set_age(mytext[line]);line++;
    emp[row].set_salary(mytext[line]);line++;
    emp[row].set_exp(mytext[line]);line++;
}

int main(){
    HashTable HLL;
    HashTabDyn HDyn;
    vector<Employee> emp;
    emp.resize(9);
    int line=0;
    for(int i=0;i<emp.size();i++){
        readData(emp,i,line);
    }

    for(int i=0;i<emp.size();i++) {HLL.insert(emp[i]);}
    HLL.print();
    cout<<"--------------------------------\n";
    HLL.remove(emp[3].get_Name());
    HLL.print();
    cout<<"--------------------------------\n";

    for(int i=0;i<emp.size();i++) HDyn.insert(emp[i]);

    HDyn.print();
    cout<<"--------------------------------\n";
    HDyn.remove("Mariam");
    HDyn.print();

    return 0;
}
```

The readData function reads the data from the .txt file and they are inserted afterwards through a loop in the vector of employee objects. After each object of the vector is inserted in the both hashtables, and the print and remove functions are executed. Note: the collision rate in this assignment is assumed to be the number of collisions divided by the number of elements in the hashtable.

Output:



```
0: Fawzy Fatma
2: Roshdy
5: Ayman Aya Abdallah
7: Mina Mariam
9: Yara
The collision rate is: 4/9 or 0.444444
--------------------------------
0: Fawzy Fatma
2: Roshdy
5: Ayman Aya Abdallah
7: Mina
9: Yara
The collision rate is: 4/8 or 0.5
--------------------------------
0: Fawzy
1: Abdallah
2: Roshdy
3: Fatma
4:
5: Ayman
6: Aya
7: Mina
8: Mariam
9: Yara
The collision rate is: 4/9 or 0.444444
--------------------------------
0: Fawzy
1: Abdallah
2: Roshdy
3: Fatma
4:
5: Ayman
6: Aya
7: Mina
8:
9: Yara
The collision rate is: 4/8 or 0.5
```

The first two segments are the hashtables done through the linked list method, while the other two are done through the dynamic array method. The hashing is done with the first letter of the name of the employee. The first one in each method is before the removal of “Mariam.” The number of elements decreases after the removal of an employee as the output shows. The names are printed alongside each other in the linked list method because of the changing method implemented when a collision occurs. For the other hashtable, linear probing is implemented to deal with the collisions.

I think that the first hashtable (the one implemented using linked lists) because it uses lists for dealing with collisions, and lists are good when it comes to strings as we discussed in the lab. Also, It groups people with the same first letter together in one list. This will make it easier to look for them even if they collide with another name because I won't have to look through the whole hashtable for them; I can just check the list. This causes smaller time complexity.