```
//HW3
//Due: 11:59PM, Friday Oct. 13
//Reminder: Midterm Exam will be held on Tuesday, Oct. 17
#include <iostream>
#include <list>
#include <map>
#include <string>
#include <tuple>
#include <iomanip>
using namespace std;

class course {
public:
        string name;//CIS554
        int section;//1
        int credits;//3
        string grade;//A-
        course() {}
        course(string n, int s, int c, string g) { name = n; section = s; credits = c; grade
= g; }

        bool operator<(const course& c) const;
        bool operator==(const course& c) const;
        float num_grade();

};
float course::num_grade() {
        map<string, float> M{
                {"A", 4.0f},
                {"A-", 3.667f},
                {"B+",  3.333f},
                {"B",   3.0f},
                {"B-" ,  2.667f},
                {"C+" ,  2.333f},
                {"C",   2.0f},
                {"C-" ,  1.667f},
                {"D",   1.0f},
                {"D-" , 0.667f},
                {"F",   0.0f}
        };
        return M[grade];
}
bool course::operator<(const course& c) const {
        return (name < c.name);
}
bool course::operator==(const course& c) const {
        return (name == c.name);

}

class DBclass {
public:
 map<int, tuple<int, float, list<course*> >* >  * pDB;
 DBclass();//default constructor
 DBclass(const initializer_list<pair<int, initializer_list<course>>>& I);//{"20222",
"CIS554", "CS#381", "MAT398", "PSY205"};

 DBclass(const DBclass& D);//Copy Constructor

 //Implement copy constructor, copy assignment, move constructor, move assignment,
destructor, RemoveFirst.
 //Also implement operator<< to allow cout << DBclassObject<<endl;

 ~DBclass();//Destructor
```

```
 void operator=(const DBclass& D);//Copy Assignment (i.e., Lvalue operator=)
 DBclass(DBclass&& D);//move constructor
 void operator=(DBclass&& D); //Move Assignment (i.e., Rvalue operator=)

 DBclass RemoveFirst();//Return a DBclass which is the same as *this,
        // but the removal of the first element of the map.
};


DBclass::DBclass() {//default constructor
        pDB = new  map<int, tuple<int, float, list<course*> >* >{};
        cout << "Constructor" << endl;
}


DBclass::DBclass(const initializer_list<pair<int, initializer_list<course>>>& I): DBclass()
{//{{"20222", "CIS554", "CS#381", "MAT398", "PSY205"}};
        for (auto& i : I) {//i is a pair
                auto semester{ i.first };//
                (*pDB)[semester] = new tuple<int, float, list<course*>>{ 0, 0.0f, {}
};//*pDB is map
                auto& R1{ *(*pDB)[semester] };//R1 is tuple
                auto& R2{ get<2>(R1) };//R2 is list
                int credit{ 0 };
                float gpa{ 0.0f };

                for (auto j : i.second) {//j is initializer_list of courses
                        R2.push_back(new course{ j });
                        gpa = (gpa * credit + j.credits * j.num_grade()) / (credit +
j.credits);
                        credit += j.credits;
                }
                get<0>(R1) = credit;
                get<1>(R1) = gpa;
        }

        cout << "Initializer List" << endl;
}


ostream& operator<<(ostream& str, const course& c);

ostream& operator<<(ostream& str, const DBclass & D);



int main() {
        course C1("CIS554", 1, 3, "A-"), C2("CSE674", 1, 3, "B+"), C3("MAT296", 8, 4, "A"),
C4("WRT205", 5, 3, "A"), C5("CSE661", 1, 3, "B+"),
                C6("PSY205", 11, 3, "A-"), C7("MAT297", 8, 4, "B+"), C8("CIS341", 1, 3, "A-
"), C9("CIS453", 1, 3, "A");

        DBclass DB1{ {20222, {C1, C2, C3}} , {20221, {C4, C5, C6}}, {20231, {C7, C8, C9}} };
        cout << "DB1: "<<endl<<DB1 << endl;

        DBclass DB2{ DB1 };
        cout << "DB2: " <<endl<<DB2 << endl;


        DBclass DB3{};
        DB3 = DB2;
        cout <<"DB3: " <<endl<< DB3 << endl;

        DB3 = DB2.RemoveFirst();//remove the first semester information from the map
        cout << DB3.pDB->size() << endl;
```

```
        cout << "DB3: " <<endl<< DB3 << endl;


        return 0;
}

ostream& operator<<(ostream& str, const DBclass& D) {

}
ostream& operator<<(ostream& str, const course& c) {
        str << "(" << c.name << " " << c.section << " " << c.credits << " " << c.grade <<
")";
        return str;
}
```