Implement *deep copy constructor* and *dropStudent()* for the *Course* class discussed during week 5.

You can only provide hand written code for *deep copy constructor* and *dropStudent()*.

Provide full source code including implementation of all of the member functions and their application in the main() via github. (provide link to the repository in the pdf)

Code:

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

using namespace std;

class Course {
public:
   Course(const string& courseName, int capacity);
   Course(const Course& other); // Deep copy constructor
   ~Course();
   Course& operator=(const Course& other); // Deep copy assignment operator
   string getCourseName() const;
   void addStudent(const string& name);
   void dropStudent(const string& name);
   string* getStudents() const;
   int getNumberOfStudents() const;

private:
   string courseName;
   string* students;
   int numberOfStudents;
   int capacity;
};

Course::Course(const string& courseName, int capacity) {
   numberOfStudents = 0;
   this->courseName = courseName;
   this->capacity = capacity;
   students = new string[capacity];
}

// Deep copy constructor
Course::Course(const Course& other) {
   courseName = other.courseName;
   capacity = other.capacity;
```

```
    numberOfStudents = other.numberOfStudents;
    students = new string[capacity];
    for (int i = 0; i < numberOfStudents; ++i) {
        students[i] = other.students[i];
    }
}

// Destructor
Course::~Course() {
    delete[] students;
}

// Deep copy assignment operator
Course& Course::operator=(const Course& other) {
    if (this == &other) {
        return *this;
    }

    delete[] students;

    courseName = other.courseName;
    capacity = other.capacity;
    numberOfStudents = other.numberOfStudents;
    students = new string[capacity];
    for (int i = 0; i < numberOfStudents; ++i) {
        students[i] = other.students[i];
    }

    return *this;
}

string Course::getCourseName() const {
    return courseName;
}

void Course::addStudent(const string& name) {
    if (numberOfStudents < capacity) {
        students[numberOfStudents] = name;
        numberOfStudents++;
    } else {
        cout << "Course is full. Cannot add more students." << endl;
    }
}

void Course::dropStudent(const string& name) {
    for (int i = 0; i < numberOfStudents; i++) {
        if (students[i] == name) {
            for (int j = i; j < numberOfStudents - 1; j++) {
                students[j] = students[j + 1];
            }
            numberOfStudents--;
            return;
        }
    }
    cout << "Student " << name << " not found in the course." << endl;
}

string* Course::getStudents() const {
    return students;
}

int Course::getNumberOfStudents() const {
    return numberOfStudents;
}

int main() {
    Course course1("Data Structures", 10);
    Course course2("Database Systems", 15);

    course1.addStudent("Irsa");
    course1.addStudent("Ayesha");
    course1.addStudent("Saleha");

    course2.addStudent("Junaid");
    course2.addStudent("Ahmed");

    cout << "Number of students in course1: " << course1.getNumberOfStudents() << "\n";
    string* students = course1.getStudents();
    for (int i = 0; i < course1.getNumberOfStudents(); i++) {
        cout << students[i] << ", ";
    }
    cout << "\nNumber of students in course2: " << course2.getNumberOfStudents() << "\n";
    students = course2.getStudents();
    for (int i = 0; i < course2.getNumberOfStudents(); i++) {
        cout << students[i] << ", ";
    }

    return 0;
}
```