```cpp
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;
class Student
{
private:
    string firstName;
    string lastName;
    string studentID;
    string phoneNumber;
    double gpa;

public:
    Student(string, string = " ", string = " ", string = " ", double = 0);
    Student();
    string getFirstName() const;
    string getLastName() const;
    string getStudentID() const;
    string getPhoneNumber() const;
    double getGPA() const;
    void setFirstName(string);
    void setLastName(string);
    void setStudentID(string);
    void setPhoneNumber(string);
    void setGPA(double);
    void input();
    void output() const;
    bool operator==(const Student&);
};
Student::Student(string a, string b , string c , string d , double e)
{
    firstName = a;
    lastName = b;
    studentID = c;
    phoneNumber = d;
    gpa = e;
}
Student::Student()
{
    firstName = " ";
    lastName = " ";
    studentID = " ";
    phoneNumber = " ";
    gpa = 0;
}
string Student::getFirstName() const
{
    return firstName;
```

```cpp
}
string Student::getLastName() const
{
    return lastName;
}
string Student::getStudentID() const
{
    return studentID;
}
string Student::getPhoneNumber() const
{
    return phoneNumber;
}
double Student::getGPA() const
{
    return gpa;
}
void Student::setFirstName(string x)
{
    firstName = x;
}
void Student::setLastName(string x)
{
    lastName = x;
}
void Student::setStudentID(string x)
{
    studentID = x;
}
void Student::setPhoneNumber(string x)
{
    phoneNumber = x;
}
void Student::setGPA(double x)
{
    gpa = x;
}
void Student::input()
{
    cout << "\n---Please enter student information---\n";
    cout << "First name: ";
    getline(cin,this->firstName);
    cout << "Last name: ";
    getline(cin,this->lastName);
    cout << "Student ID: ";
    getline(cin,this->studentID);
    cout << "Phone Number: ";
    getline(cin,this->phoneNumber);
    cout << "GPA: ";
    cin >> this->gpa;
    cin.ignore();
}
void Student::output() const
{
    string temp;
    cout << left << setw(20) << firstName << setw(20) << lastName <<
        setw(10) << studentID << setw(15) << phoneNumber <<
        setw(5) << gpa << endl;
}
bool Student::operator==(const Student& a)
{
    return (this->firstName == a.firstName && this->lastName == a.lastName
      && this->studentID == a.studentID && this->phoneNumber ==
      a.phoneNumber && this->gpa == a.gpa);
}
class Course
{
private:
    string code;
    int section;
    int capacity;
    int numStudents;
    Student* list;
public:
    Course();
    Course(string, int, int);
    ~Course();
    string getCode() const;
    int getSection() const;
    int getCapacity() const;
    int getNumStudents() const;
    void setCode(string);
    void setSection(int);
    void add(Student);
    void remove(string);

    //Display functions
    void display() const;
    void displayByFirst(string) const;
    void displayByLast(string) const;
    void displayBystudentID(string) const;
    void displayByPhone(string) const;

    //Sort functions
    void sortByFirstAsc();
    void sortByFirstDes();
    void sortByLastAsc();
    void sortByLastDes();
    void sortBystudentIDAsc();
    void sortBystudentIDDes();
    void sortByPhoneAsc();
    void sortByPhoneDes();
    void sortByGPAAsc();
    void sortByGPADes();
};

Course::Course()
{
    code = "CMPT";
    section = 1;
    capacity = 35;
    numStudents = 0;
    list = new Student[capacity];
}
Course::Course(string a, int b, int c)
{
    code = a;
    section = b;
    capacity = c;
    numStudents = 0;
    list = new Student[capacity];
}
Course::~Course()
{
    delete[] list;
}
string Course::getCode() const
{
    return this->code;
}
int Course::getSection() const
{
    return this->section;
}
int Course::getCapacity() const
{
    return this->capacity;
}
int Course::getNumStudents() const
{
    return this->numStudents;
}
void Course::setCode(string a)
{
    this->code = a;
}
void Course::setSection(int a)
```

```cpp
{
    this->section = a;
}
void Course::add(Student a)
{
    if (numStudents < capacity)
    {
        list[numStudents] = a;
        numStudents++;
        cout << "Student " << a.getFirstName() << " " << a.getLastName() << " added" << endl;
        cout << "Current number of students is: " << numStudents << endl;
    }
    else
        cout << "This course is full!!!";
}
void Course::remove(string a)
{
    int found = 0;
    for (int i = 0; i < numStudents; i++)
    {
        if (list[i].getPhoneNumber() == a)
        {
            found = 1;
            cout << "\nStudent " << list[i].getFirstName() << " " << list[i].getLastName() << " with phone number: " << a << " has been removed successfully and class list has been updated!\n";
            for (int x = i + 1; x < numStudents; x++)
            {
                list[i] = list[x];
                i++;
            }
            numStudents--;
            break;
        }
    }
    if (found != 1)
        cout << "Student not found!";
}

//Displays
void Course::display() const
{
    cout << "\n-----------Class List-----------\n";
    cout << "Course code: " << this->code << endl;
    cout << "Course section: " << this->section << endl;
    cout << "Course capacity: " << this->capacity << endl;
    cout << "Number of students: " << this->numStudents << endl;
    cout << "List of students: " << endl;
    for (int i = 0; i < numStudents; i++)
        list[i].output();
    cout << endl;
}
void Course::displayByFirst(string a) const
{
    int found = 0;
    cout << "---------------------------\n" << "Seraching for student with ⤶
       first name: " << a << "........\n";
    for (int i = 0; i < numStudents; i++)
    {
        if (list[i].getFirstName() == a)
        {
            found = 1;
            list[i].output();
        }
    }
    if (found != 1)
        cout << "Student not found!" << endl;
}

void Course::displayByLast(string a) const
{
    int found = 0;
    cout << "---------------------------\n" << "Seraching for student with ⤶
       last name: " << a << "........\n";
    for (int i = 0; i < numStudents; i++)
    {
        if (list[i].getLastName() == a)
        {
            found = 1;
            list[i].output();
        }
    }
    if (found != 1)
        cout << "Student not found!" << endl;
}

void Course::displayBystudentID(string a) const
{
    int found = 0;
    cout << "---------------------------\n" << "Seraching for student with ⤶
       Student studentID: " << a << "........\n";
    for (int i = 0; i < numStudents; i++)
    {
        if (list[i].getStudentID() == a)
        {
            found = 1;
```

```cpp
                list[i].output();
            }
        }
        if (found != 1)
            cout << "Student not found!" << endl;
    }

    void Course::displayByPhone(string a) const
    {
        int found = 0;
        cout << "--------------------------\n" << "Seraching for student with 
          Phone Number: " << a << "........\n";
        for (int i = 0; i < numStudents; i++)
        {
            if (list[i].getPhoneNumber() == a)
            {
                found = 1;
                list[i].output();
            }
        }
        if (found != 1)
            cout << "Student not found!"<<endl;
    }

    //Sorts
    void Course::sortByFirstAsc()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start+1; i < numStudents; i++)
            {
                if (list[i].getFirstName() < mStudent.getFirstName())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortByFirstDes()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getFirstName() > mStudent.getFirstName())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortByLastAsc()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getLastName() < mStudent.getLastName())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortByLastDes()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getLastName() > mStudent.getLastName())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortBystudentIDAsc()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getStudentID() < mStudent.getStudentID())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortBystudentIDDes()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getStudentID() > mStudent.getStudentID())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortByPhoneAsc()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getPhoneNumber() < mStudent.getPhoneNumber())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortByPhoneDes()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getPhoneNumber() > mStudent.getPhoneNumber())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortByGPAAsc()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getGPA() < mStudent.getGPA())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    void Course::sortByGPADes()
    {
        int mIndex;
        Student mStudent;
        for (int start = 0; start < (numStudents - 1); start++)
        {
            mIndex = start;
            mStudent = list[start];
            for (int i = start + 1; i < numStudents; i++)
            {
                if (list[i].getGPA() > mStudent.getGPA())
                {
                    mIndex = i;
                    mStudent = list[i];
                }
            }
            swap(list[mIndex], list[start]);
        }
        this->display();
    }

    int main()
    {
        string s_temp;
        Student a("Cristiano","Ronaldo", "Por001", "212-555-5555", 3.98);
        Student b("Lionel", "Messy", "Arg001", "313-555-5555", 3.99);
        Student c("Kylian", "Mbappe", "Fra001", "604-555-5555", 3.75);
        Student d("Erling", "Haaland", "Nor001", "235-555-5555", 3.51);
        Student e("Neymar", "Santos", "Bra001", "404-555-4444", 3.68);

        //Testing Class
```

```cpp
    Course z("CMPT 1209", 3, 35);
    z.add(a);
    z.add(b);
    z.add(c);
    z.add(d);
    z.add(e);

    cout << "Showing course info using Accessors" << endl;
    cout << "Course code: " << z.getCode() << endl;
    cout << "Course sectionion: " << z.getSection() << endl;
    cout << "Course capacityacity: " << z.getCapacity() << endl;
    cout << "Course occupancy: " << z.getNumStudents() << endl;

    cout << "\n------Current course information------\n";
    z.display();

    cout << "Please input a student FIRST name to locate a student by firstName name: " << endl;
    cin >> s_temp;
    z.displayByFirst(s_temp);

    cout << "Please input a student LAST name to locate a student by lastName name: " << endl;
    cin >> s_temp;
    z.displayByLast(s_temp);

    cout << "Please input a student studentID to locate a student by studentID: " << endl;
    cin >> s_temp;
    z.displayBystudentID(s_temp);

    cout << "Please input a student Phone Number to locate a student by Phone Number: " << endl;
    cin >> s_temp;
    z.displayByPhone(s_temp);

    cout << "\n-----Sorting tests-----\n" << endl;
    cout << "Press enter to begin sorting by ascending firstName names" << endl;
    cin.get();
    z.sortByFirstAsc();
    cout << "Press enter to begin sorting by descending firstName names" << endl;
    cin.get();
    z.sortByFirstDes();
    cout << "Press enter to begin sorting by ascending lastName names" << endl;
    cin.get();
    z.sortByLastAsc();
    cout << "Press enter to begin sorting by descending lastName names" << endl;
    cin.get();
    z.sortByLastDes();
    cout << "Press enter to begin sorting by ascending student studentIDs" << endl;
    cin.get();
    z.sortBystudentIDAsc();
    cout << "Press enter to begin sorting by descending student studentIDs" << endl;
    cin.get();
    z.sortBystudentIDDes();
    cout << "Press enter to begin sorting by ascending Phone Numbers" << endl;
    cin.get();
    z.sortByPhoneAsc();
    cout << "Press enter to begin sorting by descending Phone Numbers" << endl;
    cin.get();
    z.sortByPhoneDes();
    cout << "Press enter to begin sorting by ascending GPAS" << endl;
    cin.get();
    z.sortByGPAAsc();
    cout << "Press enter to begin sorting by descending GPAS" << endl;
    cin.get();
    z.sortByGPADes();

    //Removing a student using a phone number
    z.display();
    cout << "Please choose a student above and enter their phone number to drop them from the class" << endl;
    cin >> s_temp;
    z.remove(s_temp);
    z.display();
    return 0;
}
```