

```
1 #include <iostream>
2 #include <string>
3 #include <iomanip>
4 using namespace std;
5 class Student
6 {
7 private:
8     string firstName;
9     string lastName;
10    string studentID;
11    string phoneNumber;
12    double gpa;
13
14 public:
15     Student(string, string = " ", string = " ", string = " ", double = 0);
16     Student();
17     string getFirstName() const;
18     string getLastName() const;
19     string getStudentID() const;
20     string getPhoneNumber() const;
21     double getGPA() const;
22     void setFirstName(string);
23     void setLastName(string);
24     void setStudentID(string);
25     void setPhoneNumber(string);
26     void setGPA(double);
27     void input();
28     void output() const;
29     bool operator==(const Student&);
30 };
31 Student::Student(string a, string b , string c , string d , double e)
32 {
33     firstName = a;
34     lastName = b;
35     studentID = c;
36     phoneNumber = d;
37     gpa = e;
38 }
39 Student::Student()
40 {
41     firstName = " ";
42     lastName = " ";
43     studentID = " ";
44     phoneNumber = " ";
45     gpa = 0;
46 }
47 string Student::getFirstName() const
48 {
49     return firstName;
```

```
50 }
51 string Student::getLastName() const
52 {
53     return lastName;
54 }
55 string Student::getStudentID() const
56 {
57     return studentID;
58 }
59 string Student::getPhoneNumber() const
60 {
61     return phoneNumber;
62 }
63 double Student::getGPA() const
64 {
65     return gpa;
66 }
67 void Student::setFirstName(string x)
68 {
69     firstName = x;
70 }
71 void Student::setLastName(string x)
72 {
73     lastName = x;
74 }
75 void Student::setStudentID(string x)
76 {
77     studentID = x;
78 }
79 void Student::setPhoneNumber(string x)
80 {
81     phoneNumber = x;
82 }
83 void Student::setGPA(double x)
84 {
85     gpa = x;
86 }
87 void Student::input()
88 {
89     cout << "\n---Please enter student information---\n";
90     cout << "First name: ";
91     getline(cin, this->firstName);
92     cout << "Last name: ";
93     getline(cin, this->lastName);
94     cout << "Student ID: ";
95     getline(cin, this->studentID);
96     cout << "Phone Number: ";
97     getline(cin, this->phoneNumber);
98     cout << "GPA: ";
```

```
99     cin >> this->gpa;
100     cin.ignore();
101 }
102 void Student::output() const
103 {
104     string temp;
105     cout << left << setw(20) << firstName << setw(20) << lastName <<
106         setw(10) << studentID << setw(15) << phoneNumber <<
107         setw(5) << gpa << endl;
108 }
109 bool Student::operator==(const Student& a)
110 {
111     return (this->firstName == a.firstName && this->lastName == a.lastName &
112             && this->studentID == a.studentID && this->phoneNumber ==
113             a.phoneNumber && this->gpa == a.gpa);
114 }
115 class Course
116 {
117 private:
118     string code;
119     int section;
120     int capacity;
121     int numStudents;
122     Student* list;
123 public:
124     Course();
125     Course(string, int, int);
126     ~Course();
127     string getCode() const;
128     int getSection() const;
129     int getCapacity() const;
130     int getNumStudents() const;
131     void setCode(string);
132     void setSection(int);
133     void add(Student);
134     void remove(string);
135
136     //Display functions
137     void display() const;
138     void displayByFirst(string) const;
139     void displayByLast(string) const;
140     void displayByStudentID(string) const;
141     void displayByPhone(string) const;
142
143     //Sort functions
144     void sortByFirstAsc();
145     void sortByFirstDes();
146     void sortByLastAsc();
147     void sortByLastDes();
```

```
146     void sortByStudentIDAsc();
147     void sortByStudentIDDes();
148     void sortByPhoneAsc();
149     void sortByPhoneDes();
150     void sortByGPAAsc();
151     void sortByGPADes();
152 };
153
154 Course::Course()
155 {
156     code = "CMPT";
157     section = 1;
158     capacity = 35;
159     numStudents = 0;
160     list = new Student[capacity];
161 }
162 Course::Course(string a, int b, int c)
163 {
164     code = a;
165     section = b;
166     capacity = c;
167     numStudents = 0;
168     list = new Student[capacity];
169 }
170 Course::~~Course()
171 {
172     delete[] list;
173 }
174 string Course::getCode() const
175 {
176     return this->code;
177 }
178 int Course::getSection() const
179 {
180     return this->section;
181 }
182 int Course::getCapacity() const
183 {
184     return this->capacity;
185 }
186 int Course::getNumStudents() const
187 {
188     return this->numStudents;
189 }
190 void Course::setCode(string a)
191 {
192     this->code = a;
193 }
194 void Course::setSection(int a)
```

```
195 {
196     this->section = a;
197 }
198 void Course::add(Student a)
199 {
200     if (numStudents < capacity)
201     {
202         list[numStudents] = a;
203         numStudents++;
204         cout << "Student " << a.getFirstName() << " " << a.getLastName() << " added" << endl;
205         cout << "Current number of students is: " << numStudents << endl;
206     }
207     else
208         cout << "This course is full!!!";
209 }
210 void Course::remove(string a)
211 {
212     int found = 0;
213     for (int i = 0; i < numStudents; i++)
214     {
215         if (list[i].getPhoneNumber() == a)
216         {
217             found = 1;
218             cout << "\nStudent " << list[i].getFirstName() << " " << list[i].getLastName() << " with phone number: " << a << " has been removed successfully and class list has been updated! \n";
219             for (int x = i + 1; x < numStudents; x++)
220             {
221                 list[i] = list[x];
222                 i++;
223             }
224             numStudents--;
225             break;
226         }
227     }
228     if (found != 1)
229         cout << "Student not found!";
230 }
231
232 //Displays
233 void Course::display() const
234 {
235     cout << "\n-----Class List-----\n";
236     cout << "Course code: " << this->code << endl;
237     cout << "Course section: " << this->section << endl;
238     cout << "Course capacity: " << this->capacity << endl;
239     cout << "Number of students: " << this->numStudents << endl;
```

```
240     cout << "List of students: " << endl;
241     for (int i = 0; i < numStudents; i++)
242         list[i].output();
243     cout << endl;
244 }
245 void Course::displayByFirst(string a) const
246 {
247     int found = 0;
248     cout << "-----\n" << "Seraching for student with ↵
        first name: " << a << ".....\n";
249     for (int i = 0; i < numStudents; i++)
250     {
251         if (list[i].getFirstName() == a)
252         {
253             found = 1;
254             list[i].output();
255         }
256     }
257     if (found != 1)
258         cout << "Student not found!" << endl;
259 }
260
261 void Course::displayByLast(string a) const
262 {
263     int found = 0;
264     cout << "-----\n" << "Seraching for student with ↵
        last name: " << a << ".....\n";
265     for (int i = 0; i < numStudents; i++)
266     {
267         if (list[i].getLastName() == a)
268         {
269             found = 1;
270             list[i].output();
271         }
272     }
273     if (found != 1)
274         cout << "Student not found!" << endl;
275 }
276
277 void Course::displayBystudentID(string a) const
278 {
279     int found = 0;
280     cout << "-----\n" << "Seraching for student with ↵
        Student studentID: " << a << ".....\n";
281     for (int i = 0; i < numStudents; i++)
282     {
283         if (list[i].getStudentID() == a)
284         {
285             found = 1;
```

```
286         list[i].output();
287     }
288 }
289 if (found != 1)
290     cout << "Student not found!" << endl;
291 }
292
293 void Course::displayByPhone(string a) const
294 {
295     int found = 0;
296     cout << "-----\n" << "Seraching for student with ↗
        Phone Number: " << a << ".....\n";
297     for (int i = 0; i < numStudents; i++)
298     {
299         if (list[i].getPhoneNumber() == a)
300         {
301             found = 1;
302             list[i].output();
303         }
304     }
305     if (found != 1)
306         cout << "Student not found!"<<endl;
307 }
308
309 //Sorts
310 void Course::sortByFirstAsc()
311 {
312     int mIndex;
313     Student mStudent;
314     for (int start = 0; start < (numStudents - 1); start++)
315     {
316         mIndex = start;
317         mStudent = list[start];
318         for (int i = start+1; i < numStudents; i++)
319         {
320             if (list[i].getFirstName() < mStudent.getFirstName())
321             {
322                 mIndex = i;
323                 mStudent = list[i];
324             }
325         }
326         swap(list[mIndex], list[start]);
327     }
328     this->display();
329 }
330
331 void Course::sortByFirstDes()
332 {
333     int mIndex;
```

```
334     Student mStudent;
335     for (int start = 0; start < (numStudents - 1); start++)
336     {
337         mIndex = start;
338         mStudent = list[start];
339         for (int i = start + 1; i < numStudents; i++)
340         {
341             if (list[i].getFirstName() > mStudent.getFirstName())
342             {
343                 mIndex = i;
344                 mStudent = list[i];
345             }
346         }
347         swap(list[mIndex], list[start]);
348     }
349     this->display();
350 }
351
352 void Course::sortByLastAsc()
353 {
354     int mIndex;
355     Student mStudent;
356     for (int start = 0; start < (numStudents - 1); start++)
357     {
358         mIndex = start;
359         mStudent = list[start];
360         for (int i = start + 1; i < numStudents; i++)
361         {
362             if (list[i].getLastName() < mStudent.getLastName())
363             {
364                 mIndex = i;
365                 mStudent = list[i];
366             }
367         }
368         swap(list[mIndex], list[start]);
369     }
370     this->display();
371 }
372
373 void Course::sortByLastDes()
374 {
375     int mIndex;
376     Student mStudent;
377     for (int start = 0; start < (numStudents - 1); start++)
378     {
379         mIndex = start;
380         mStudent = list[start];
381         for (int i = start + 1; i < numStudents; i++)
382         {
```



```
383         if (list[i].getLastName() > mStudent.getLastName())
384         {
385             mIndex = i;
386             mStudent = list[i];
387         }
388     }
389     swap(list[mIndex], list[start]);
390 }
391 this->display();
392 }
393
394 void Course::sortByStudentIDAsc()
395 {
396     int mIndex;
397     Student mStudent;
398     for (int start = 0; start < (numStudents - 1); start++)
399     {
400         mIndex = start;
401         mStudent = list[start];
402         for (int i = start + 1; i < numStudents; i++)
403         {
404             if (list[i].getStudentID() < mStudent.getStudentID())
405             {
406                 mIndex = i;
407                 mStudent = list[i];
408             }
409         }
410         swap(list[mIndex], list[start]);
411     }
412     this->display();
413 }
414
415 void Course::sortByStudentIDDes()
416 {
417     int mIndex;
418     Student mStudent;
419     for (int start = 0; start < (numStudents - 1); start++)
420     {
421         mIndex = start;
422         mStudent = list[start];
423         for (int i = start + 1; i < numStudents; i++)
424         {
425             if (list[i].getStudentID() > mStudent.getStudentID())
426             {
427                 mIndex = i;
428                 mStudent = list[i];
429             }
430         }
431         swap(list[mIndex], list[start]);
```

```
432     }
433     this->display();
434 }
435
436 void Course::sortByPhoneAsc()
437 {
438     int mIndex;
439     Student mStudent;
440     for (int start = 0; start < (numStudents - 1); start++)
441     {
442         mIndex = start;
443         mStudent = list[start];
444         for (int i = start + 1; i < numStudents; i++)
445         {
446             if (list[i].getPhoneNumber() < mStudent.getPhoneNumber())
447             {
448                 mIndex = i;
449                 mStudent = list[i];
450             }
451         }
452         swap(list[mIndex], list[start]);
453     }
454     this->display();
455 }
456
457 void Course::sortByPhoneDes()
458 {
459     int mIndex;
460     Student mStudent;
461     for (int start = 0; start < (numStudents - 1); start++)
462     {
463         mIndex = start;
464         mStudent = list[start];
465         for (int i = start + 1; i < numStudents; i++)
466         {
467             if (list[i].getPhoneNumber() > mStudent.getPhoneNumber())
468             {
469                 mIndex = i;
470                 mStudent = list[i];
471             }
472         }
473         swap(list[mIndex], list[start]);
474     }
475     this->display();
476 }
477
478 void Course::sortByGPAAsc()
479 {
480     int mIndex;
```

```
481     Student mStudent;
482     for (int start = 0; start < (numStudents - 1); start++)
483     {
484         mIndex = start;
485         mStudent = list[start];
486         for (int i = start + 1; i < numStudents; i++)
487         {
488             if (list[i].getGPA() < mStudent.getGPA())
489             {
490                 mIndex = i;
491                 mStudent = list[i];
492             }
493         }
494         swap(list[mIndex], list[start]);
495     }
496     this->display();
497 }
498
499 void Course::sortByGPADes()
500 {
501     int mIndex;
502     Student mStudent;
503     for (int start = 0; start < (numStudents - 1); start++)
504     {
505         mIndex = start;
506         mStudent = list[start];
507         for (int i = start + 1; i < numStudents; i++)
508         {
509             if (list[i].getGPA() > mStudent.getGPA())
510             {
511                 mIndex = i;
512                 mStudent = list[i];
513             }
514         }
515         swap(list[mIndex], list[start]);
516     }
517     this->display();
518 }
519
520 int main()
521 {
522     string s_temp;
523     Student a("Cristiano", "Ronaldo", "Por001", "212-555-5555", 3.98);
524     Student b("Lionel", "Messy", "Arg001", "313-555-5555", 3.99);
525     Student c("Kylian", "Mbappe", "Fra001", "604-555-5555", 3.75);
526     Student d("Erling", "Haaland", "Nor001", "235-555-5555", 3.51);
527     Student e("Neymar", "Santos", "Bra001", "404-555-4444", 3.68);
528
529     //Testing Class
```

```
530     Course z("CMPT 1209", 3, 35);
531     z.add(a);
532     z.add(b);
533     z.add(c);
534     z.add(d);
535     z.add(e);
536
537     cout << "Showing course info using Accessors" << endl;
538     cout << "Course code: " << z.getCode() << endl;
539     cout << "Course sectionion: " << z.getSection() << endl;
540     cout << "Course capacityacity: " << z.getCapacity() << endl;
541     cout << "Course occupancy: " << z.getNumStudents() << endl;
542
543     cout << "\n-----Current course information-----\n";
544     z.display();
545
546     cout << "Please input a student FIRST name to locate a student by      ↗
547         firstName name: " << endl;
548     cin >> s_temp;
549     z.displayByFirst(s_temp);
550
551     cout << "Please input a student LAST name to locate a student by      ↗
552         lastName name: " << endl;
553     cin >> s_temp;
554     z.displayByLast(s_temp);
555
556     cout << "Please input a student studentID to locate a student by      ↗
557         studentID: " << endl;
558     cin >> s_temp;
559     z.displayBystudentID(s_temp);
560
561     cout << "Please input a student Phone Number to locate a student by    ↗
562         Phone Number: " << endl;
563     cin >> s_temp;
564     z.displayByPhone(s_temp);
565
566     cout << "\n-----Sorting tests-----\n" << endl;
567     cout << "Press enter to begin sorting by ascending firstName names" << ↗
568         endl;
569     cin.get();
570     z.sortByFirstAsc();
571     cout << "Press enter to begin sorting by descending firstName names" ↗
572         << endl;
573     cin.get();
574     z.sortByFirstDes();
575     cout << "Press enter to begin sorting by ascending lastName names" << ↗
576         endl;
577     cin.get();
578     z.sortByLastAsc();
```

```
572     cout << "Press enter to begin sorting by descending lastName names" << ↵
        endl;
573     cin.get();
574     z.sortByLastDes();
575     cout << "Press enter to begin sorting by ascending student studentIDs" ↵
        << endl;
576     cin.get();
577     z.sortBystudentIDAsc();
578     cout << "Press enter to begin sorting by descending student ↵
        studentIDs" << endl;
579     cin.get();
580     z.sortBystudentIDDes();
581     cout << "Press enter to begin sorting by ascending Phone Numbers" << ↵
        endl;
582     cin.get();
583     z.sortByPhoneAsc();
584     cout << "Press enter to begin sorting by descending Phone Numbers" << ↵
        endl;
585     cin.get();
586     z.sortByPhoneDes();
587     cout << "Press enter to begin sorting by ascending GPAS" << endl;
588     cin.get();
589     z.sortByGPAAsc();
590     cout << "Press enter to begin sorting by descending GPAS" << endl;
591     cin.get();
592     z.sortByGPADes();
593
594     //Removing a student using a phone number
595     z.display();
596     cout << "Please choose a student above and enter their phone number to ↵
        drop them from the class" << endl;
597     cin >> s_temp;
598     z.remove(s_temp);
599     z.display();
600     return 0;
601 }
```