```cpp
/*
 *  - domesticstudent.cpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// make sure to include the hpp file for the declarations; this file will be
// actually implementing the functions in the class
#include "domesticstudent.hpp"

using namespace std;

// ****************** DomesticStudent Class ****************** //

// define implementations for the get and set functions
string DomesticStudent::getProvince() const
{
    return this->province;
}

void DomesticStudent::setProvince(string s)
{
    this->province = s;
}

// define implementations for custom operators

ostream& operator << (ostream& outs, const DomesticStudent &object)
{
    outs << setw(14) << left << object.getFirstName() << " " << setw(18) <<  left <<
    object.getLastName();
    outs << setw(10) << left << object.getProvince();
    outs << setw(6) << left << object.getCGPA();
    outs << setw(4) << left << object.getResearchScore() << endl;
    return outs;
}

istream& operator >> (istream &ins,  DomesticStudent &object)
{
    string s_tmp;
    getline(ins, s_tmp, ',');
    object.setFirstName(s_tmp);
    getline(ins, s_tmp, ',');
    object.setLastName(s_tmp);
    getline(ins, s_tmp, ',');
    object.setProvince(s_tmp);
    getline(ins, s_tmp, ',');
    object.setCGPA(atof(s_tmp.c_str()));
    getline(ins, s_tmp, ',');
    object.setResearchScore(atoi(s_tmp.c_str()));
    return ins;
}

// default constructor, initialize strings to ""
DomesticStudent::DomesticStudent()
{
    this->setProvince("");
}

// ****************** Friend functions of DomesticStudent Class ****************** //

int compareCGPA(DomesticStudent student1, DomesticStudent student2)
{
    return compareTwoNumbers(student1.getCGPA(), student2.getCGPA());
}

int compareResearchScore(DomesticStudent student1, DomesticStudent student2)
```

```
{
    return compareTwoNumbers(student1.getResearchScore(), student2.getResearchScore());
}

int compareFirstName(DomesticStudent student1, DomesticStudent student2)
{
    return compareTwoStrings(student1.getFirstName(), student2.getFirstName());
}

int compareLastName(DomesticStudent student1, DomesticStudent student2)
{
    return compareTwoStrings(student1.getLastName(), student2.getLastName());
}

```

```
/*
 *  - domesticstudent.hpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// multiple inclusion prevention
#ifndef DOMESTICSTUDENT_HPP
#define DOMESTICSTUDENT_HPP

#include <iostream>
#include <iomanip>
#include <string>
#include "student.hpp"

using namespace std;

// this class shares properties with the Student class above since it's a child
// class but it also has new members unique to DomesticStudent only
class DomesticStudent : public Student
{
public:
    // declare the get and set functions as public so that the main program can
    // access them
    string getProvince() const;
    void setProvince(string s);
    // overload the insertion operator to print out the object's information
    friend ostream& operator << (ostream& outs, const DomesticStudent &object);
    // overload the extraction operator to copy a file line into the class
    friend istream& operator >> (istream &ins,  DomesticStudent &object);
    // define friend functions for this class - friend functions can access
    // any of the private members in this class but they don't belong to a
    // particular class
    friend int compareCGPA(DomesticStudent student1, DomesticStudent student2);
    friend int compareResearchScore(DomesticStudent student1, DomesticStudent student2);
    friend int compareFirstName(DomesticStudent student1, DomesticStudent student2);
    friend int compareLastName(DomesticStudent student1, DomesticStudent student2);
    // create a constructor
    DomesticStudent();

private:
    // declare the actual variables as private so that the main program can't
    // access them and overwrite by accident
    string province;
};

#endif

```

```cpp
/*
 *  - Internationalstudent.cpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// make sure to include the hpp file for the declarations; this file will be
// actually implementing the functions in the class
#include "internationalstudent.hpp"

using namespace std;

// ****************** InternationalStudent Class ****************** //

// define implementations for the get and set functions
string InternationalStudent::getCountry() const
{
    return this->country;
}

void InternationalStudent::setCountry(string c)
{
    this->country = c;
}

ToeflScore InternationalStudent::getToeflScore() const
{
    ToeflScore t;
    t.setReading(this->toeflScore.getReading());
    t.setListening(this->toeflScore.getListening());
    t.setSpeaking(this->toeflScore.getSpeaking());
    t.setWriting(this->toeflScore.getWriting());
    return t;
}

void InternationalStudent::setToeflScore(ToeflScore t)
{
    this->toeflScore.setReading(t.getReading());
    this->toeflScore.setListening(t.getListening());
    this->toeflScore.setSpeaking(t.getSpeaking());
    this->toeflScore.setWriting(t.getWriting());
}

void InternationalStudent::setToeflScore(int reading, int listening, int speaking, int
writing)
{
    this->toeflScore.setReading(reading);
    this->toeflScore.setListening(listening);
    this->toeflScore.setSpeaking(speaking);
    this->toeflScore.setWriting(writing);
}

// this is the start of the redundant functions found in ToeflScore

int InternationalStudent::getReading() const
{
    return this->toeflScore.getReading();
}

void InternationalStudent::setReading(int r)
{
    this->toeflScore.setReading(r);
}

int InternationalStudent::getListening() const
{
    return this->toeflScore.getListening();
```

```
69  }
70
71  void InternationalStudent::setListening(int l)
72  {
73      this->toeflScore.setListening(l);
74  }
75
76  int InternationalStudent::getSpeaking() const
77  {
78      return this->toeflScore.getSpeaking();
79  }
80
81  void InternationalStudent::setSpeaking(int s)
82  {
83      this->toeflScore.setSpeaking(s);
84  }
85
86  int InternationalStudent::getWriting() const
87  {
88      return this->toeflScore.getWriting();
89  }
90
91  void InternationalStudent::setWriting(int w)
92  {
93      this->toeflScore.setWriting(w);
94  }
95
96  int InternationalStudent::getTotalScore() const
97  {
98      int total = 0;
99      total += this->toeflScore.getReading();
100     total += this->toeflScore.getListening();
101     total += this->toeflScore.getSpeaking();
102     total += this->toeflScore.getWriting();
103     return total;
104 }
105
106 // define implementations for custom operators
107
108 ostream& operator << (ostream& outs, const InternationalStudent &object)
109 {
110     outs << setw(14) << left << object.getFirstName() << " " << setw(18) << left <<
        object.getLastName();
111     outs << setw(10) << left << object.getCountry();
112     outs << setw(6) << left << object.getCGPA();
113     outs << setw(4) << left << object.getResearchScore();
114     outs << setw(4) << left << object.getReading();
115     outs << setw(4) << left << object.getListening();
116     outs << setw(4) << left << object.getSpeaking();
117     outs << setw(4) << left << object.getWriting();
118     outs << object.getTotalScore() << endl;
119     return outs;
120 }
121
122 istream& operator >> (istream &ins, InternationalStudent &object)
123 {
124     string s_tmp;
125     getline(ins, s_tmp, ',');
126     object.setFirstName(s_tmp);
127     getline(ins, s_tmp, ',');
128     object.setLastName(s_tmp);
129     getline(ins, s_tmp, ',');
130     object.setCountry(s_tmp);
131     getline(ins, s_tmp, ',');
132     object.setCGPA(atof(s_tmp.c_str()));
133     getline(ins, s_tmp, ',');
134     object.setResearchScore(atoi(s_tmp.c_str()));
135     getline(ins, s_tmp, ',');
136     object.setReading(atoi(s_tmp.c_str()));
```

```
137         getline(ins, s_tmp, ',');
138         object.setListening(atoi(s_tmp.c_str()));
139         getline(ins, s_tmp, ',');
140         object.setSpeaking(atoi(s_tmp.c_str()));
141         getline(ins, s_tmp, ',');
142         object.setWriting(atoi(s_tmp.c_str()));
143         return ins;
144     }
145
146     // default constructor, initialize strings to ""
147     InternationalStudent::InternationalStudent()
148     {
149         this->setCountry("");
150     }
151
152     // ****************** Friend functions of DomesticStudent Class ****************** //
153
154     int compareCGPA(InternationalStudent student1, InternationalStudent student2)
155     {
156         return compareTwoNumbers(student1.getCGPA(), student2.getCGPA());
157     }
158
159     int compareResearchScore(InternationalStudent student1, InternationalStudent student2)
160     {
161         return compareTwoNumbers(student1.getResearchScore(), student2.getResearchScore());
162     }
163
164     int compareFirstName(InternationalStudent student1, InternationalStudent student2)
165     {
166         return compareTwoStrings(student1.getFirstName(), student2.getFirstName());
167     }
168
169     int compareLastName(InternationalStudent student1, InternationalStudent student2)
170     {
171         return compareTwoStrings(student1.getLastName(), student2.getLastName());
172     }
173
```

```cpp
/*
 *  - internationalstudent.hpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// multiple inclusion prevention
#ifndef INTERNATIONALSTUDENT_HPP
#define INTERNATIONALSTUDENT_HPP

#include <iostream>
#include <iomanip>
#include <string>
#include "student.hpp"
#include "toeflscore.hpp"

using namespace std;

// this class shares properties with the Student class above since it's a child
// class but it also has new members unique to InternationalStudent only
class InternationalStudent : public Student
{
public:
    // declare the get and set functions as public so that the main program can
    // access them
    string getCountry() const;
    void setCountry(string c);
    ToeflScore getToeflScore() const;
    void setToeflScore(ToeflScore t);
    void setToeflScore(int reading, int listening, int speaking, int writing);
    // redundant access to the getReading, getWriting, etc. found in ToeflScore
    int getReading() const;
    void setReading(int r);
    int getListening() const;
    void setListening(int l);
    int getSpeaking() const;
    void setSpeaking(int s);
    int getWriting() const;
    void setWriting(int w);
    int getTotalScore() const;
    // overload the insertion operator to print out the object's information
    friend ostream& operator << (ostream& outs, const InternationalStudent &object);
    // overload the extraction operator to copy a file line into the class
    friend istream& operator >> (istream &ins, InternationalStudent &object);
    // define friend functions for this class - friend functions can access
    // any of the private members in this class but they don't belong to a
    // particular class
    friend int compareCGPA(InternationalStudent student1, InternationalStudent student2);
    friend int compareResearchScore(InternationalStudent student1, InternationalStudent
    student2);
    friend int compareFirstName(InternationalStudent student1, InternationalStudent
    student2);
    friend int compareLastName(InternationalStudent student1, InternationalStudent
    student2);
    // create a constructor
    InternationalStudent();

private:
    // declare the actual variables as private so that the main program can't
    // access them and overwrite by accident
    ToeflScore toeflScore;
    string country;
};

#endif

```

```cpp
/*
 *  - main.cpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// include standard libraries
#include <iostream>
#include <fstream>
#include <sstream>
#include <cstdlib>
// include user defined libraries
#include "student.hpp"
#include "domesticstudent.hpp"
#include "toeflscore.hpp"
#include "internationalstudent.hpp"

using namespace std;

DomesticStudent* sortBy(DomesticStudent* unsortedDomesticStudents, int num, string type,
 string order)
{
    int currentIndex;
    int ascending;

    if (type != "F" && type != "L" && type != "C" && type != "R") return NULL;//return
    -1;
    if (order == "ascending") ascending = 1;
    else ascending = -1;

    DomesticStudent* sortedDomesticStudents = new DomesticStudent[num];
    int* usedIndex = new int[num];

    for (int i = 0; i < num; i++)
    {
        for (int j = 0; j < num; j++)
        {
            if (usedIndex[j] != 1)
            {
                currentIndex = j;
                break;
            }
        }
        for (int j = 0; j < num; j++)
        {
            if (usedIndex[j] == 1) continue;
            if (((type == "F") && (compareFirstName(unsortedDomesticStudents[
            currentIndex], unsortedDomesticStudents[j]) * ascending > 0))
            ||  ((type == "L") && (compareLastName(unsortedDomesticStudents[currentIndex
            ], unsortedDomesticStudents[j]) * ascending > 0))
            ||  ((type == "C") && (compareCGPA(unsortedDomesticStudents[currentIndex],
            unsortedDomesticStudents[j]) * ascending > 0))
            ||  ((type == "R") && (compareResearchScore(unsortedDomesticStudents[
            currentIndex], unsortedDomesticStudents[j]) * ascending) > 0))
            {
                currentIndex = j;
            }
        }
        sortedDomesticStudents[i] = unsortedDomesticStudents[currentIndex];
        usedIndex[currentIndex] = 1;
    }

    delete [] usedIndex;
    return sortedDomesticStudents;
}

InternationalStudent* sortBy(InternationalStudent* unsortedInternationalStudents, int
```

```
    num, string type, string order)
64  {
65      int currentIndex;
66      int ascending;
67
68      if (type != "F" && type != "L" && type != "C" && type != "R") return NULL;
69      if (order == "ascending") ascending = 1;
70      else ascending = -1;
71
72      InternationalStudent* sortedInternationalStudents = new InternationalStudent[num];
73      int* usedIndex = new int[num];
74
75      int k = 0;
76
77      for (int i = 0; i < num; i++)
78      {
79          for (int j = 0; j < num; j++)
80          {
81              if (usedIndex[j] != 1)
82              {
83                  currentIndex = j;
84                  break;
85              }
86          }
87          for (int j = 0; j < num; j++)
88          {
89              if (usedIndex[j] == 1) continue;
90              if (((type == "F") && compareFirstName(unsortedInternationalStudents[
                currentIndex], unsortedInternationalStudents[j]) == ascending)
91              ||  ((type == "L") && compareLastName(unsortedInternationalStudents[
                currentIndex], unsortedInternationalStudents[j]) == ascending)
92              ||  ((type == "C") && compareCGPA(unsortedInternationalStudents[currentIndex
                ], unsortedInternationalStudents[j]) == ascending)
93              ||  ((type == "R") && compareResearchScore(unsortedInternationalStudents[
                currentIndex], unsortedInternationalStudents[j]) == ascending))
94              {
95                  currentIndex = j;
96              }
97          }
98          if (unsortedInternationalStudents[currentIndex].getReading() > 20
99          &&  unsortedInternationalStudents[currentIndex].getListening() > 20
100         &&  unsortedInternationalStudents[currentIndex].getSpeaking() > 20
101         &&  unsortedInternationalStudents[currentIndex].getWriting() > 20
102         &&  unsortedInternationalStudents[currentIndex].getTotalScore() > 93)
103         {
104             sortedInternationalStudents[k++] = unsortedInternationalStudents[
                currentIndex];
105         }
106         usedIndex[currentIndex] = 1;
107     }
108
109     delete [] usedIndex;
110
111     return sortedInternationalStudents;
112 }
113
114 DomesticStudent* sortBy(DomesticStudent* unsortedDomesticStudents, int num, string type)
115 {
116     return sortBy(unsortedDomesticStudents, num, type, "ascending");
117 }
118
119 InternationalStudent* sortBy(InternationalStudent* unsortedDomesticStudents, int num,
    string type)
120 {
121     return sortBy(unsortedDomesticStudents, num, type, "ascending");
122 }
123
124 ifstream fileOpen(string fname)
125 {
```

```
126         ifstream file(fname);
127         if(!file.is_open())
128         {
129             cout << "Unable to open file " << fname << ". Terminating" << endl;
130             exit(-1);
131         }
132         return file;
133     }
134
135     int countLines(ifstream& file)
136     {
137         int i = 0;
138         string line;
139         while(getline(file, line)) i++;
140         return i;
141     }
142
143
144     int main()
145     {
146         string studentSelection, sortingSelection, line;
147         cout << "Please select which group you would like to sort:\nDomestic Students (D)
            International Students(I)  Overall Sort (O)\n";
148         cin >> studentSelection;
149         cout << "Please enter what you would like to sort by:\nFirst Name (F)  Last Name
            (L)  CGPA (C)  Research Score (R)\n";
150         cin >> sortingSelection;
151
152
153
154         if (studentSelection == "D" || studentSelection == "O")
155         {
156             ifstream domesticFile = fileOpen("domestic-stu.txt");
157
158             int numDomesticStudents = countLines(domesticFile) - 1;
159
160             DomesticStudent* unsortedDomesticStudents = new DomesticStudent[
                numDomesticStudents];
161
162             domesticFile.clear();
163             domesticFile.seekg(0, ios::beg);
164             getline(domesticFile, line);
165
166             int i = 0;
167             while(getline(domesticFile, line))
168             {
169                 istringstream ss(line);
170                 ss >> unsortedDomesticStudents[i++];
171             }
172
173             DomesticStudent* sortedDomesticStudents = sortBy(unsortedDomesticStudents,
                numDomesticStudents, sortingSelection);
174
175             if (sortedDomesticStudents == NULL)
176             {
177                 cout << "Your input was invalid. Please re run this program\n";
178                 return -1;
179             }
180
181             cout << "\n\nSorted DomesticStudent Array:\n\n";
182             for (int i = 0; i < numDomesticStudents; i++) cout << sortedDomesticStudents[i];
183
184             domesticFile.close();
185             delete [] unsortedDomesticStudents;
186             delete [] sortedDomesticStudents;
187         }
188         else if (studentSelection == "I" || studentSelection == "O")
189         {
190             ifstream internationalFile = fileOpen("international-stu.txt");
```

```
191
192             int numInternationalStudents = countLines(internationalFile) - 1;
193
194             InternationalStudent* unsortedInternationalStudents = new InternationalStudent[
                numInternationalStudents];
195
196             internationalFile.clear();
197             internationalFile.seekg(0, ios::beg);
198             getline(internationalFile, line);
199
200             int i = 0;
201             while(getline(internationalFile, line))
202             {
203                 istringstream ss(line);
204                 ss >> unsortedInternationalStudents[i++];
205             }
206
207             InternationalStudent* sortedInternationalStudents = sortBy(
                unsortedInternationalStudents, numInternationalStudents, sortingSelection);
208
209             if (sortedInternationalStudents == NULL)
210             {
211                 cout << "Your input was invalid. Please re run this program\n";
212                 return -1;
213             }
214
215             cout << "\n\nSorted InternationalStudent Array:\n\n";
216             for (int i = 0; i < numInternationalStudents; i++)
217             {
218                 if (sortedInternationalStudents[i].getFirstName() == "") break;
219                 else cout << sortedInternationalStudents[i];
220             }
221
222             internationalFile.close();
223             delete [] unsortedInternationalStudents;
224             delete [] sortedInternationalStudents;
225         }
226         else
227         {
228             cout << "Your input was invalid. Please re run this program.\n";
229             return -1;
230         }
231
232         return 0;
233     }
234
```

```cpp
/*
 *  - student.cpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// make sure to include the hpp file for the declarations; this file will be
// actually implementing the functions in the class
#include "student.hpp"

using namespace std;

// ****************** Student Class ****************** //

// define implementations for the get and set functions
string Student::getFirstName() const
{
    return this->firstName;
}

void Student::setFirstName(string s)
{
    this->firstName = s;
}

string Student::getLastName() const
{
    return this->lastName;
}

void Student::setLastName(string s)
{
    this->lastName = s;
}

float Student::getCGPA() const
{
    return this->CGPA;
}

void Student::setCGPA(float c)
{
    this->CGPA = c;
}

int Student::getResearchScore() const
{
    return this->researchScore;
}

void Student::setResearchScore(int rs)
{
    this->researchScore = rs;
}

// default constructor, initialize ints/floats to 0.0 and strings to ""
Student::Student()
{
    this->setFirstName("");
    this->setLastName("");
    this->setCGPA(0.0);
    this->setResearchScore(0);
}

// ****************** Common functions used by both DomesticStudent and
InternationalStudent ****************** //

```

```
69  int compareTwoNumbers(int n1, int n2)
70  {
71      if (n1 > n2) return 1;
72      if (n1 < n2) return -1;
73      return 0;
74  }
75
76  int compareTwoNumbers(float n1, float n2)
77  {
78      if (n1 > n2) return 1;
79      if (n1 < n2) return -1;
80      return 0;
81  }
82
83  int compareTwoStrings(string s1, string s2)
84  {
85      for (int i = 0; i < s1.size(); i++) if (s1[i] >= 'a' && s1[i] <= 'z') s1[i] -= ('a'-
        'A');
86      for (int i = 0; i < s2.size(); i++) if (s2[i] >= 'a' && s2[i] <= 'z') s2[i] -= ('a'-
        'A');
87      return s1.compare(s2);
88  }
89
```

```cpp
/*
 *  - student.hpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// multiple inclusion prevention
#ifndef STUDENT_HPP
#define STUDENT_HPP

#include <iostream>
#include <string>

using namespace std;

// main student class, this is the same for all students, domestic and
// international
class Student
{
public:
    // declare the get and set functions as public so that the main program can
    // access them
    string getFirstName() const;
    void setFirstName(string s);
    string getLastName() const;
    void setLastName(string s);
    float getCGPA() const;
    void setCGPA(float c);
    int getResearchScore() const;
    void setResearchScore(int rs);
    // create a constructor
    Student();

private:
    // declare the actual variables as private so that the main program can't
    // access them and overwrite by accident
    string firstName;
    string lastName;
    float CGPA;
    int researchScore;
};

// functions outside of class but are common to both types of students

// returns a 1 if n1 > n2, 0 if n1 == n2, -1 if n1 < n2
int compareTwoNumbers(int n1, int n2);

// returns a 1 if n1 > n2, 0 if n1 == n2, -1 if n1 < n2
int compareTwoNumbers(float n1, float n2);

// returns a 1 if s1 > s2, 0 if s1 == s2, -1 if s1 < s2
int compareTwoStrings(string s1, string s2);

#endif

```

```cpp
/*
 *  - toeflscore.cpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// make sure to include the hpp file for the declarations; this file will be
// actually implementing the functions in the class
#include "toeflscore.hpp"

using namespace std;

// ***************** ToeflScore Class ***************** //

// define implementations for the get and set functions
int ToeflScore::getReading() const
{
    return this->reading;
}

void ToeflScore::setReading(int r)
{
    this->reading = r;
}

int ToeflScore::getListening() const
{
    return this->listening;
}

void ToeflScore::setListening(int l)
{
    this->listening = l;
}

int ToeflScore::getSpeaking() const
{
    return this->speaking;
}

void ToeflScore::setSpeaking(int s)
{
    this->speaking = s;
}

int ToeflScore::getWriting() const
{
    return this->writing;
}

void ToeflScore::setWriting(int w)
{
    this->writing = w;
}

int ToeflScore::getTotalScore() const
{
    int total = 0;
    total += this->reading;
    total += this->listening;
    total += this->speaking;
    total += this->writing;
    return total;
}

// default constructor, initialize ints/floats to 0.0
ToeflScore::ToeflScore()
```

```
70  {
71      this->setReading(0);
72      this->setListening(0);
73      this->setSpeaking(0);
74      this->setWriting(0);
75  }
76
```

```cpp
/*
 *  - toeflscore.hpp -
 * ENSC 251 - Lab Assignment 2
 * May 31, 2019
 * Judd Foster
 * 301377893
 */

// multiple inclusion prevention
#ifndef TOEFLSCORE_HPP
#define TOEFLSCORE_HPP

#include <iostream>
#include <string>

using namespace std;

// this class holds the reading, writing, speaking, etc. values for the
// InternationalStudent class, it makes it more organized
class ToeflScore
{
public:
    // declare the get and set functions as public so that the main program can
    // access them
    int getReading() const;
    void setReading(int r);
    int getListening() const;
    void setListening(int l);
    int getSpeaking() const;
    void setSpeaking(int s);
    int getWriting() const;
    void setWriting(int w);
    int getTotalScore() const;
    // create a constructor
    ToeflScore();

private:
    // declare the actual variables as private so that the main program can't
    // access them and overwrite by accident
    int reading;
    int listening;
    int speaking;
    int writing;
};

#endif

```