File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\Date.h

```
 1
 2 #ifndef FINAL_PROJECT_DATE_H
 3 #define FINAL_PROJECT_DATE_H
 4 #include "ostream"
 5
 6 using namespace std;
 7
 8 class Date
 9 {
10     friend ostream& operator << (ostream&, const Date&); // method overloading
11     friend istream& operator >> (istream&, Date&);
12
13 public:
14
15     // default constructor
16     Date();
17
18     // specific constructor
19     Date(int d, int m, int y);
20
21     // destructor
22     ~Date();
23
24     // setters
25     void setDay(int d);
26     void setMonth(int m);
27     void setYear(int y);
28     void setDmy(int d, int m, int y);
29
30     // getters
31     int getDay() const;
32     int getMonth() const;
33     int getYear() const;
34
35     // printf
36     void print() const;
37
38     // Overloading operators
39     bool operator<(const Date& otherDate) const;
40     bool operator>(const Date& otherDate) const;
41     bool operator==(const Date& otherDate) const;
42     bool operator<=(const Date& otherDate) const;
43     bool operator>=(const Date& otherDate) const;
```

```
44      bool operator!=(const Date& otherDate) const;
45
46      // Overloading math operations
47      Date operator++();
48      Date operator++(int);
49      Date operator--();
50      Date operator--(int);
51
52  private:
53
54      // data members
55      int day;
56      int month;
57      int year;
58  };
59
60  #endif
61
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\Time.h

```
 1 //
 2 // Created by hycwy on 4/9/2022.
 3 //
 4
 5 #ifndef FINAL_PROJECT_TIME_H
 6 #define FINAL_PROJECT_TIME_H
 7 #include "ostream"
 8
 9 using namespace std;
10
11 class Time {
12
13     friend ostream& operator << (ostream&, const Time&); // method overloading
14     friend istream& operator >> (istream&, Time&);
15
16 public:
17
18     // default constructor
19     Time();
20
21     // specific constructor
22     Time(int s, int m, int h);
23
24     // setters
25     void setSec(int s);
26     void setMin(int m);
27     void setHour(int h);
28     void setSmh(int s, int m, int h);
29
30     // getters
31     int getSec() const;
32     int getMin() const;
33     int getHour() const;
34
35     // printf
36     void print() const;
37
38     // Overloading operators
39     bool operator<(const Time& otherTime) const;
40     bool operator>(const Time& otherTime) const;
41     bool operator==(const Time& otherTime) const;
42     bool operator<=(const Time& otherTime) const;
43     bool operator>=(const Time& otherTime) const;
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\Time.h

```
44      bool operator!=(const Time& otherTime) const;
45
46      // Overloading math operations
47      Time operator++();
48      Time operator++(int);
49      Time operator--();
50      Time operator--(int);
51
52  private:
53
54      // data members
55      int sec;
56      int min;
57      int hour;
58  };
59
60
61  #endif //FINAL_PROJECT_TIME_H
62
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\Date.cpp

```cpp
#include <iostream>
#include "Date.h"

using namespace std;

// default constructor
Date::Date()
{
    day = 1;
    month = 1;
    year = 2000;
}

// specific constructor
Date::Date(int d, int m, int y)
{
    day = d;
    month = m;
    year = y;
}

// destructor
Date::~Date() {

}

// setters
void Date::setDay(int d)
{
    day = d;
}

void Date::setMonth(int m)
{
    month = m;
}

void Date::setYear(int y)
{
    year = y;
}

void Date::setDmy(int d, int m, int y)
```

```
{
    day = d;
    month = m;
    year = y;
}

// getters
int Date::getDay() const
{
    return day;
}

int Date::getMonth() const
{
    return month;
}

int Date::getYear() const
{
    return year;
}

// printf
void Date::print() const
{
    cout << day << "/"
         << month << "/"
         << year;
}

// ostream operator
ostream& operator << (ostream& osObject, const Date& date1) //creating an instance of a date class, i.e. date1
{
    osObject << date1.day
             << "/" << date1.month
             << "/" << date1.year;
    return osObject;
}

// istream operator
istream& operator >> (istream& isObject, Date& date1)
{
    isObject >> date1.day >> date1.month >> date1.year;
```

```
 87     return isObject;
 88 }
 89
 90 // Overloading "equal to" operator
 91 bool Date::operator==(const Date& otherDate) const
 92 {
 93     if (day == otherDate.day && month == otherDate.month
 94         && year == otherDate.year)
 95         return true;
 96     else
 97         return false;
 98 }
 99
100 // Overloading "not equal to" operator
101 bool Date::operator!=(const Date& otherDate) const
102 {
103     return !(*this == otherDate);
104 }
105
106 // Overloading "less than or equal to" operator
107 bool Date::operator<=(const Date& otherDate) const
108 {
109     return (*this < otherDate || *this == otherDate);
110 }
111
112 // Overloading "less than" operator
113 bool Date::operator<(const Date& otherDate) const
114 {
115     if ((year < otherDate.year) ||
116         (year == otherDate.year && month < otherDate.month) ||
117         (year == otherDate.year && month == otherDate.month && day < otherDate.day))
118         return true;
119     else
120         return false;
121 }
122
123 // Overloading "greater than or equal to" operator
124 bool Date::operator>=(const Date& otherDate) const
125 {
126     return !(*this < otherDate);
127 }
128
129 // Overloading "greater than" operator
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\Date.cpp

```
130 bool Date::operator>(const Date& otherDate) const
131 {
132     return !(*this <= otherDate);
133 }
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\main.cpp

```cpp
#include <iostream>
#include <fstream>
#include <iomanip>
#include <sstream>
#include <algorithm>
#include "vector"
#include "string"
#include "Date.h"
#include "Time.h"
#include "AirQuality.h"
using namespace std;

void getData(ifstream &inFile, vector<string>& fields, vector<AirQuality>& airQualityCollection);
Date processDate(string inputStr);
Time processTime(string inputStr);
double calculateAvgTemp(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection);
double calculateRelHumidity(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection);
double calculateAbsHumidity(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection);
void displayTempAndRelHumidity(Date dateInput, Time timeInput, double &currentTemp, double &currentRelHumidity, vector<AirQuality>&
airQualityCollection);
double displayMaxTemp(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection);
double displayMaxRelHumidity(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection);
double displayMaxAbsHumidity(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection);
void findTempHigherThanAvg(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection, vector<AirQuality>& tempHigherThanAvg);
void findRelHumidHigherThanAvg(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection, vector<AirQuality>&
relHumidHigherThanAvg);
void findAbsHumidHigherThanAvg(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection, vector<AirQuality>&
absHumidHigherThanAvg);
void showWelcomeMsg();
void showMenu();
int  userChoice();
bool validMonth(int monthInput);
void doUserChoice(int option, vector<AirQuality>& airQualityCollection);
Time getValidTime();
int getValidInputMonth();
int getValidInputYear();
int getValidInputDay(int inputYear, int inputMonth);
void doCalOrDis(int option, int yearInput, int monthInput,vector<AirQuality>& airQualityCollection);
void doCalThanAvg(int option, int yearInput, int monthInput, vector<AirQuality>& airQualityCollection, vector<AirQuality>&
absHumidHigherThanAvg);
void doFindTempAndHum(Date inputDate, Time inputTime, double &currentTemp, double &currentRelHumidity, vector<AirQuality>&
airQualityCollection);

```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\main.cpp

```
39
40 //int main()
41 //{
42 //    ifstream myFile("AirQualityUCI.csv");
43 //
44 //    if (!myFile.is_open())
45 //    {
46 //        cout << "File not found." << endl;
47 //        return 1;
48 //    }
49 //
50 //    vector<string> fields;
51 //    vector<AirQuality> airQualityCollection;
52 //    getData(myFile, fields, airQualityCollection);
53 //    showWelcomeMsg();
54 //    showMenu();
55 //    int res = userChoice();
56 //    int EXIT = -1;
57 //    while (res != EXIT)
58 //    {
59 //        doUserChoice(res, airQualityCollection);
60 //        cout << endl;
61 //        showMenu();
62 //        res = userChoice();
63 //        if (res == EXIT)
64 //        {
65 //            break;
66 //        }
67 //    }
68 //    return 0;
69 //}
70
71 int main()
72 {
73     ifstream myFile("AirQualityUCI.csv");
74
75     if (!myFile.is_open())
76     {
77         cout << "File not found." << endl;
78         return 1;
79     }
80
81     vector<string> fields;
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\main.cpp

```
 82     vector<AirQuality> airQualityCollection;
 83
 84     // Test: getData()
 85     getData(myFile, fields, airQualityCollection);
 86     // Expected: 9357, Actual: 9357
 87     cout << "Get data, size is: " << airQualityCollection.size() << endl;
 88
 89     // Test: calculateAvgTemp()
 90     // Expected: 14.3908, Actual: 14.3908
 91     cout << "avg temp is: " << calculateAvgTemp(4, 3, airQualityCollection) << endl;
 92     // Test: calculateRelHumidity()
 93     // Expected: 50.1708, Actual: 50.1708
 94     cout << "avg relHumidity is: " << calculateRelHumidity(4, 3, airQualityCollection) << endl;
 95     // Test: calculateAbsHumidity()
 96     // Expected: 0.789413, Actual: 0.789413
 97     cout << "abs relHumidity is: " << calculateAbsHumidity(4, 3, airQualityCollection) << endl;
 98
 99     // Test: displayTempAndRelHumidity()
100     // Expected: temp:13.6, relative humid:48.9, Actual: temp:13.6, relative humid:48.9
101     Date d1(10, 3, 4);
102     Time t1(0, 0, 18);
103     double currentTemp, currentRelHumidity;
104     displayTempAndRelHumidity(d1, t1, currentTemp, currentRelHumidity, airQualityCollection);
105     cout << "At that date and time, temp is: " << currentTemp << ", rel humid is: " << currentRelHumidity << endl;
106
107     // Test: displayMaxTemp()
108     // Expected: 29.3, Actual: 29.3
109     cout << "max temp is: " << displayMaxTemp(4, 3, airQualityCollection) << endl;
110     // Test: displayMaxRelHumidity()
111     // Expected: 83.2, Actual: 83.2
112     cout << "max rel is: " << displayMaxRelHumidity(4, 3, airQualityCollection) << endl;
113     // Test: displayMaxAbsHumidity()
114     // Expected: 1.0945, Actual: 1.0945
115     cout << "max abs is: " << displayMaxAbsHumidity(4, 3, airQualityCollection) << endl;
116
117     // Test: findTempHigherThanAvg()
118     // Expected: size 239, Actual: size 239
119     vector<AirQuality> tempHigherThanAvg;
120     findTempHigherThanAvg(4, 3, airQualityCollection, tempHigherThanAvg);
121     cout << "Higher than avg temp num: " << tempHigherThanAvg.size() << endl;
122
123     // Test: findRelHumidHigherThanAvg()
124     // Expected: size 285, Actual: size 285
```

```
125     vector<AirQuality>relHumidHigherThanAvg;
126     findRelHumidHigherThanAvg(4, 3, airQualityCollection, relHumidHigherThanAvg);
127     cout << "Higher than avg rel num: " << relHumidHigherThanAvg.size() << endl;
128 
129     // Test: findAbsHumidHigherThanAvg()
130     // Expected: size 255, Actual: size 255
131     vector<AirQuality>absHumidHigherThanAvg;
132     findAbsHumidHigherThanAvg(4, 3, airQualityCollection, absHumidHigherThanAvg);
133     cout << "Higher than avg abs: " << absHumidHigherThanAvg.size() << endl;
134 
135     return 0;
136 
137 }
138 
139 void getData(ifstream &inFile, vector<string>& fields, vector<AirQuality>& airQualityCollection)
140 {
141     static const int DATE_IDX = 0;
142     static const int TIME_IDX = 1;
143     static const int TEMP_IDX = 12;
144     static const int REL_HUMID_IDX = 13;
145     static const int ABS_HUMID_IDX = 14;
146 
147     string strs;
148     string delimiter = ",";
149     size_t pos;
150     // skip the first line
151     getline(inFile, strs);
152     // read each line, store as strs
153     while (getline(inFile, strs))
154     {
155         size_t size = strs.size() - 1;
156         // skip empty line (',,,,')
157         if (strs[0] == ',')
158         {
159             break;
160         }
161         for (int i = 0; i < size; ++i)
162         {
163             pos = strs.find(delimiter, i);
164             if (pos < size)
165             {
166                 string token = strs.substr(i, pos - i);
167                 fields.push_back(token);
```

```
168                 i = pos + delimiter.size() - 1;
169             }
170         }
171 
172         // create date object with element in fields vector (index = 0)
173         Date dateVal = processDate(fields.at(DATE_IDX));
174         // create time object with element in fields vector (index = 1)
175         Time timeVal = processTime(fields.at(TIME_IDX));
176         // convert other parameters from string to double
177         double temperature = stod(fields.at(TEMP_IDX));
178         double relHumid = stod(fields.at(REL_HUMID_IDX));
179         double absHumid = stod(fields.at(ABS_HUMID_IDX));
180         // create AirQuality object, store in airQualityCollection vector
181         AirQuality airQualityVal(dateVal, timeVal, temperature, relHumid, absHumid);
182         airQualityCollection.push_back(airQualityVal);
183         fields.clear();
184     }
185 
186     inFile.close();
187 }
188 
189 Date processDate(string inputStr)
190 {
191     string delimiter = "/";
192     size_t pos;
193     string strs = inputStr + delimiter;
194     size_t size = strs.size();
195     vector<int> dateValues;
196     // split the input string to 3 substrings, convert to int
197     for (int i = 0; i < size; ++i)
198     {
199         pos = strs.find(delimiter, i);
200         if (pos < size)
201         {
202             string token = strs.substr(i, pos - i);
203             int tokenNum = stoi(token);
204             dateValues.push_back(tokenNum);
205             i = pos + delimiter.size() - 1;
206         }
207     }
208 
209     static const int DAY_IDX = 1;
210     static const int MONTH_IDX = 0;
```

```
211     static const int YEAR_IDX = 2;
212     // create date object with elements in dateValues vector
213     Date dateVal(dateValues.at(DAY_IDX), dateValues.at(MONTH_IDX), dateValues.at(YEAR_IDX));
214     return dateVal;
215 }
216 
217 Time processTime(string inputStr)
218 {
219     string delimiter = ":";
220     size_t pos;
221     string strs = inputStr + delimiter;
222     size_t size = strs.size();
223     vector<int> timeValues;
224     // split the input string to 3 substrings, convert to int
225     for (int i = 0; i < size; ++i)
226     {
227         pos = strs.find(delimiter, i);
228         if (pos < size)
229         {
230             string token = strs.substr(i, pos - i);
231             int tokenNum = stoi(token);
232             timeValues.push_back(tokenNum);
233             i = pos + delimiter.size() - 1;
234         }
235     }
236 
237     static const int HOUR_IDX = 0;
238     static const int MIN_IDX = 1;
239     static const int SEC_IDX = 2;
240     // create Time object with elements in timeVal vector
241     Time timeVal(timeValues.at(SEC_IDX), timeValues.at(MIN_IDX), timeValues.at(HOUR_IDX));
242     return timeVal;
243 }
244 
245 double calculateAvgTemp(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection)
246 {
247     static const double MISS_VALUE = -200;
248     double totalTemp = 0;
249     int count = 0;
250     for (int i = 0; i < airQualityCollection.size(); i++){
251         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput){
252             if (airQualityCollection[i].getTemp() != MISS_VALUE){
253                 totalTemp += airQualityCollection[i].getTemp();
```

```
254                 count++;
255             }
256         }
257     }
258     double avgTemp = totalTemp / count;
259 
260     if (count == 0){
261         return 0;
262     }
263     else{
264         return avgTemp;
265     }
266 }
267 
268 double calculateRelHumidity(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection)
269 {
270     static const double MISS_VALUE = -200;
271     double totalHumidity = 0;
272     int count = 0;
273     for (int i = 0; i < airQualityCollection.size(); i++){
274         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput){
275             if (airQualityCollection[i].getRelativeHumidity() != MISS_VALUE){
276                 totalHumidity += airQualityCollection[i].getRelativeHumidity();
277                 count++;
278             }
279         }
280     }
281     double avgHumidity = totalHumidity / count;
282 
283     if (count == 0){
284         return 0;
285     }
286     else{
287         return avgHumidity;
288     }
289 }
290 
291 
292 double calculateAbsHumidity(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection)
293 {
294     static const double MISS_VALUE = -200;
295     double totalHumidity = 0;
296     int count = 0;
```

```
297     for (int i = 0; i < airQualityCollection.size(); i++){
298         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput){
299             if (airQualityCollection[i].getAbsHumidity() != MISS_VALUE){
300                 totalHumidity += airQualityCollection[i].getAbsHumidity();
301                 count++;
302             }
303         }
304     }
305     double avgHumidity = totalHumidity / count;
306 
307     if (count == 0){
308         return 0;
309     }
310     else{
311         return avgHumidity;
312     }
313 }
314 
315 void displayTempAndRelHumidity(Date dateInput, Time timeInput, double &currentTemp, double &currentRelHumidity, vector<AirQuality>&
    airQualityCollection)
316 {
317     for (int i = 0; i < airQualityCollection.size(); i++){
318         if (dateInput == airQualityCollection[i].getDate() && timeInput == airQualityCollection[i].getTime()){
319             currentTemp = airQualityCollection[i].getTemp();
320             currentRelHumidity = airQualityCollection[i].getRelativeHumidity();
321             return;
322         }
323     }
324     currentTemp = 0;
325     currentRelHumidity = 0;
326 }
327 
328 double displayMaxTemp(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection)
329 {
330     static const double THRESHOLD = -100;
331     double maxTemp = THRESHOLD;
332     for (int i = 0; i < airQualityCollection.size(); i++){
333         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput) {
334             maxTemp = max(maxTemp, airQualityCollection[i].getTemp());
335         }
336     }
337     return maxTemp;
338 }
```

```
339
340 double displayMaxRelHumidity(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection)
341 {
342     static const double THRESHOLD = -100;
343     double maxRelHumidity = THRESHOLD;
344     for (int i = 0; i < airQualityCollection.size(); i++){
345         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput) {
346             maxRelHumidity = max(maxRelHumidity, airQualityCollection[i].getRelativeHumidity());
347         }
348     }
349     return maxRelHumidity;
350 }
351
352 double displayMaxAbsHumidity(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection)
353 {
354     static const double THRESHOLD = -100;
355     double maxAbsHumidity = THRESHOLD;
356     for (int i = 0; i < airQualityCollection.size(); i++){
357         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput) {
358             maxAbsHumidity = max(maxAbsHumidity, airQualityCollection[i].getAbsHumidity());
359         }
360     }
361     return maxAbsHumidity;
362 }
363
364 void findTempHigherThanAvg(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection, vector<AirQuality>& tempHigherThanAvg)
365 {
366     double avgTemp = calculateAvgTemp(yearInput, monthInput, airQualityCollection);
367     // if avg = 0 -> cannot find data corresponding to monthInput
368     if (avgTemp == 0)
369     {
370         return;
371     }
372     for (int i = 0; i < airQualityCollection.size(); i++){
373         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput){
374             if (airQualityCollection[i].getTemp() > avgTemp){
375                 tempHigherThanAvg.push_back(airQualityCollection[i]);
376             }
377         }
378     }
379 }
380
381 void findRelHumidHigherThanAvg(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection, vector<AirQuality>&
```

```
381 relHumidHigherThanAvg)
382 {
383     double avgRelHumid = calculateRelHumidity(yearInput, monthInput, airQualityCollection);
384     // if avg = 0 -> cannot find data corresponding to monthInput
385     if (avgRelHumid == 0)
386     {
387         return;
388     }
389     for (int i = 0; i < airQualityCollection.size(); i++){
390         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput){
391             if (airQualityCollection[i].getRelativeHumidity() > avgRelHumid){
392                 relHumidHigherThanAvg.push_back(airQualityCollection[i]);
393             }
394         }
395     }
396 }
397
398 void findAbsHumidHigherThanAvg(int yearInput, int monthInput, vector<AirQuality>& airQualityCollection, vector<AirQuality>&
    absHumidHigherThanAvg)
399 {
400     double avgAbsHumid = calculateAbsHumidity(yearInput, monthInput, airQualityCollection);
401     // if avg = 0 -> cannot find data corresponding to monthInput
402     if (avgAbsHumid == 0)
403     {
404         return;
405     }
406     for (int i = 0; i < airQualityCollection.size(); i++){
407         if (airQualityCollection[i].getDate().getYear() == yearInput && airQualityCollection[i].getDate().getMonth() == monthInput){
408             if (airQualityCollection[i].getAbsHumidity() > avgAbsHumid){
409                 absHumidHigherThanAvg.push_back(airQualityCollection[i]);
410             }
411         }
412     }
413 }
414
415 void showWelcomeMsg()
416 {
417     cout << endl;
418     cout << "Welcome to the Air Quality Processor!" << endl;
419     cout << endl;
420 }
421
422 void showMenu()
```

```
423 {
424     cout << "********************************************************************************" << endl;
425     cout << "*                                  MENU                                        *" << endl;
426     cout << "* 0.  Display the average temperature for a certain month.                     *" << endl;
427     cout << "* 1.  Display the average relative humidity for a certain month.               *" << endl;
428     cout << "* 2.  Display the average absolute humidity temperature for a certain month. *" << endl;
429     cout << "* 3.  Display the temperature and temperature at a certain date and time.     *" << endl;
430     cout << "* 4.  Display the highest temperature for a certain month.                    *" << endl;
431     cout << "* 5.  Display the highest relative humidity value for a certain month.        *" << endl;
432     cout << "* 6.  Display the highest absolute humidity for a certain month.              *" << endl;
433     cout << "* 7.  Display temperature higher than the average for a certain month.        *" << endl;
434     cout << "* 8.  Display relative humidity higher than the average for a certain month. *" << endl;
435     cout << "* 9. Display absolute humidity higher than the average for a certain month. *" << endl;
436     cout << "*                                                                              *" << endl;
437     cout << "********************************************************************************" << endl;
438 }
439
440 int userChoice()
441 {
442     cout << endl;
443     cout << "--------------------------------------------------------------------------------" << endl;
444     cout << "Please select your option by enter the No." << endl
445          << "Enter -1 when you finished." << endl;
446     int MAX_OPTION = 10;
447     bool selecting = true;
448     while (selecting)
449     {
450         int userInput;
451         cin >> userInput;
452         while (cin.fail())
453         {
454             cin.clear();
455             int const IGNORE_SIZE = 9999999;
456             cin.ignore(IGNORE_SIZE, '\n');
457             cout << "ERROR: " << endl;
458             cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
459             showMenu();
460             cin >> userInput;
461         }
462         if (userInput >= -1 && userInput <= MAX_OPTION)
463         {
464             selecting = false;
465             return userInput;
```

```
466         }
467         else
468         {
469             cout << "ERROR: " << endl;
470             cout << "Invalid option. Option out of range. Please Re-enter." << endl;
471             showMenu();
472         }
473     }
474     return -1;
475 }
476 
477 bool validMonth(int monthInput)
478 {
479     int MIN_MONTH = 1;
480     int MAX_MONTH = 12;
481     return (monthInput >= MIN_MONTH && monthInput <= MAX_MONTH);
482 }
483 
484 bool validTime(int h, int m, int s)
485 {
486     int MIN_TIME = 0;
487     int MAX_HOUR = 24;
488     int MAX_MIN_SEC = 59;
489     bool hourValidity = ((h >= MIN_TIME) && (h <= MAX_HOUR));
490     bool minValidity = ((m >= MIN_TIME) && (m <= MAX_MIN_SEC));
491     bool secValidity = ((s >= MIN_TIME ) && (s <= MAX_MIN_SEC));
492     return (hourValidity && minValidity && secValidity);
493 }
494 
495 Time getValidTime()
496 {
497     bool invalid = true;
498     while (invalid)
499     {
500         int inputHour, inputMin, inputSec;
501         cout << "Please enter the value of hour." << endl;
502         cin >> inputHour;
503         while (cin.fail())
504         {
505             cin.clear();
506             int const IGNORE_SIZE = 9999999;
507             cin.ignore(IGNORE_SIZE, '\n');
508             cout << "ERROR: " << endl;
```

```
509             cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
510             cin >> inputHour;
511         }
512         cout << "Please enter the value of minutes." << endl;
513         cin >> inputMin;
514         while (cin.fail())
515         {
516             cin.clear();
517             int const IGNORE_SIZE = 9999999;
518             cin.ignore(IGNORE_SIZE, '\n');
519             cout << "ERROR: " << endl;
520             cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
521             cin >> inputMin;
522         }
523         cout << "Please enter the value of seconds." << endl;
524         cin >> inputSec;
525         while (cin.fail())
526         {
527             cin.clear();
528             int const IGNORE_SIZE = 9999999;
529             cin.ignore(IGNORE_SIZE, '\n');
530             cout << "ERROR: " << endl;
531             cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
532             cin >> inputSec;
533         }
534         if (validTime(inputHour, inputMin, inputSec))
535         {
536             invalid = false;
537             return Time(inputSec, inputMin, inputHour);
538         }
539         else
540         {
541             cout << "Invalid combination, please Re-enter." << endl;
542         }
543     }
544     return Time();
545 }
546
547 int getValidInputYear()
548 {
549     int yearInput;
550     cout << "Please enter a year: " << endl;
551     cin >> yearInput;
```

```
552     while (cin.fail())
553     {
554         cin.clear();
555         int const IGNORE_SIZE = 9999999;
556         cin.ignore(IGNORE_SIZE, '\n');
557         cout << "ERROR: " << endl;
558         cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
559         cin >> yearInput;
560     }
561     return yearInput;
562 }
563
564 int getValidInputMonth()
565 {
566     int monthInput;
567     cout << "Please enter a month: " << endl;
568     cin >> monthInput;
569     while (cin.fail())
570     {
571         cin.clear();
572         int const IGNORE_SIZE = 9999999;
573         cin.ignore(IGNORE_SIZE, '\n');
574         cout << "ERROR: " << endl;
575         cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
576         cin >> monthInput;
577     }
578     while (!validMonth(monthInput)) {
579         cout << "Month should be in range of [1, 12], inclusive" << endl;
580         cout << "Please re-enter." << endl;
581         cin >> monthInput;
582         while (cin.fail()) {
583             cin.clear();
584             int const IGNORE_SIZE = 9999999;
585             cin.ignore(IGNORE_SIZE, '\n');
586             cout << "ERROR: " << endl;
587             cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
588             cin >> monthInput;
589         }
590     }
591     return monthInput;
592 }
593
594 bool validDay(int inputYear, int inputMonth, int inputDay) {
```

```
595     int validDay[] = {31,28,31,30,31,30,31,31,30,31,30,31};
596     if ((inputYear % 4 == 0) && (inputYear % 100 != 0) || (inputYear % 400 == 0))
597     {
598         int februaryIndex = 2;
599         validDay[februaryIndex] = 29;
600     }
601     int SHIFT = 1;
602     return ((inputDay > 0) && (inputDay <= (validDay[(inputMonth) - SHIFT])));
603 }
604
605 int getValidInputDay(int inputYear, int inputMonth) {
606     int dayInput;
607     cout << "Please enter a day: " << endl;
608     cin >> dayInput;
609     while (cin.fail())
610     {
611         cin.clear();
612         int const IGNORE_SIZE = 9999999;
613         cin.ignore(IGNORE_SIZE, '\n');
614         cout << "ERROR: " << endl;
615         cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
616         cin >> dayInput;
617     }
618     while (!validDay(inputYear,inputMonth,dayInput)) {
619         cout << "Input day is not valid, there is no " << dayInput << " in the month of " << inputMonth << endl;
620         cout << "Please re-enter." << endl;
621         cin >> dayInput;
622         while (cin.fail()) {
623             cin.clear();
624             int const IGNORE_SIZE = 9999999;
625             cin.ignore(IGNORE_SIZE, '\n');
626             cout << "ERROR: " << endl;
627             cout << "Invalid option. Only number accepted. Please Re-enter." << endl;
628             cin >> dayInput;
629         }
630     }
631     return dayInput;
632 }
633
634 void doUserChoice(int option, vector<AirQuality>& airQualityCollection)
635 {
636     int EXIT = -1;
637     if (option == EXIT)
```

```
638     {
639         return;
640         // exit from the function
641     }
642     int yearInput = getValidInputYear();
643     int monthInput = getValidInputMonth();
644     if (option == 3)
645     {
646         int dayInput = getValidInputDay(yearInput, monthInput);
647         Date curDay = Date(dayInput, monthInput, yearInput);
648         Time curTime = getValidTime();
649         double curTemp, curRelHumidity;
650         doFindTempAndHum(curDay, curTime, curTemp, curRelHumidity, airQualityCollection);
651     }
652     else
653     {
654         if (option == 0 || option == 1 || option == 2 ||
655             option == 4 || option == 5 || option == 6)
656         {
657 
658             doCalOrDis(option, yearInput, monthInput, airQualityCollection);
659         }
660         else
661         {
662             vector<AirQuality> res;
663             doCalThanAvg(option, yearInput, monthInput, airQualityCollection, res);
664         }
665     }
666 }
667 
668 void doCalOrDis(int option, int yearInput, int monthInput,vector<AirQuality>& airQualityCollection)
669 {
670     double res;
671     int NO_RECORD_CAL = 0;
672     int NO_RECORD_DIS = -100;
673     switch (option) {
674         case 0:
675             res = calculateAvgTemp(yearInput, monthInput, airQualityCollection);
676             if (res == NO_RECORD_CAL)
677             {
678                 cout << "No records found." << endl;
679             }
680             else
```

```
681             {
682                 cout << "The average temperature is: " << res << endl;
683             }
684             break;
685         case 1:
686             res = calculateRelHumidity(yearInput, monthInput, airQualityCollection);
687             if (res == NO_RECORD_CAL)
688             {
689                 cout << "No records found." << endl;
690             }
691             else
692             {
693                 cout << "The average relative humidity is: " << res << endl;
694             }
695             break;
696         case 2:
697             res = calculateAbsHumidity(yearInput, monthInput, airQualityCollection);
698             if (res == NO_RECORD_CAL)
699             {
700                 cout << "No records found." << endl;
701             }
702             else
703             {
704                 cout << "The average absolute humidity is: " << res << endl;
705             }
706             break;
707         case 4:
708             res = displayMaxTemp(yearInput, monthInput, airQualityCollection);
709             if (res == NO_RECORD_DIS)
710             {
711                 cout << "No records found." << endl;
712             }
713             else
714             {
715                 cout << "The highest temperature is: " << res << endl;
716             }
717             break;
718         case 5:
719             res = displayMaxRelHumidity(yearInput, monthInput, airQualityCollection);
720             if (res == NO_RECORD_DIS)
721             {
722                 cout << "No records found." << endl;
723             }
```

```
724             else
725             {
726                 cout << "The highest relative humidity is: " << res << endl;
727             }
728             break;
729         case 6:
730             res = displayMaxAbsHumidity(yearInput, monthInput, airQualityCollection);
731             if (res == NO_RECORD_DIS)
732             {
733                 cout << "No records found." << endl;
734             }
735             else
736             {
737                 cout << "The highest absolute humidity is: " << res << endl;
738             }
739             break;
740     }
741 }
742
743 void doCalThanAvg(int option, int yearInput, int monthInput, vector<AirQuality>& airQualityCollection, vector<AirQuality>& res)
744 {
745     switch (option) {
746         case 7:
747             findTempHigherThanAvg(yearInput, monthInput, airQualityCollection, res);
748             if (res.size() == 0)
749             {
750                 cout << "No records found." << endl;
751             }
752             else
753             {
754                 cout << "The dates and times when temperature is higher than the average are listed below: " << endl;
755                 for (int i = 0; i < res.size(); i++)
756                 {
757                     cout << "Date:\t" << res[i].getDate() << "\t" << "Time:\t" << res[i].getTime() << endl;
758                 }
759             }
760         case 8:
761             findRelHumidHigherThanAvg(yearInput, monthInput, airQualityCollection, res);
762             if (res.size() == 0)
763             {
764                 cout << "No records found." << endl;
765             }
766             else
```

```
767             {
768                 cout << "The dates and times when temperature is higher than the average are listed below: " << endl;
769                 for (int i = 0; i < res.size(); i++)
770                 {
771                     cout << "Date:\t" << res[i].getDate() << "\t" << "Time:\t" << res[i].getTime() << endl;
772                 }
773             }
774         case 9:
775             findAbsHumidHigherThanAvg(yearInput, monthInput, airQualityCollection, res);
776             if (res.size() == 0)
777             {
778                 cout << "No records found." << endl;
779             }
780             else
781             {
782                 cout << "The dates and times when temperature is higher than the average are listed below: " << endl;
783                 for (int i = 0; i < res.size(); i++)
784                 {
785                     cout << "Date:\t" << res[i].getDate() << "\t" << "Time:\t" << res[i].getTime() << endl;
786                 }
787             }
788     }
789 }
790
791 void doFindTempAndHum(Date inputDate, Time inputTime, double &currentTemp, double &currentRelHumidity, vector<AirQuality>&
    airQualityCollection)
792 {
793     displayTempAndRelHumidity(inputDate, inputTime, currentTemp, currentRelHumidity,airQualityCollection);
794     if (currentTemp == 0 && currentRelHumidity == 0)
795     {
796         cout << "No records found." << endl;
797     }
798     else
799     {
800         cout << "At that date and time, temp is: " << currentTemp << ", rel humid is: " << currentRelHumidity << endl;
801     }
802 }
803
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\Time.cpp

```cpp
#include <iostream>
#include "Time.h"

using namespace std;

// default constructor
Time::Time()
{
    sec = 1;
    min = 1;
    hour = 1;
}

// specific constructor
Time::Time(int s, int m, int h)
{
    sec = s;
    min = m;
    hour = h;
}

// setters
void Time::setSec(int s)
{
    sec = s;
}

void Time::setMin(int m)
{
    min = m;
}

void Time::setHour(int h)
{
    hour = h;
}

void Time::setSmh(int s, int m, int h)
{
    sec = s;
    min = m;
    hour = h;
}
```

```
44
45 // getters
46 int Time::getSec() const
47 {
48     return sec;
49 }
50
51 int Time::getMin() const
52 {
53     return min;
54 }
55
56 int Time::getHour() const
57 {
58     return hour;
59 }
60
61 // printf
62 void Time::print() const
63 {
64     cout << sec << ":"
65          << min << ":"
66          << hour;
67 }
68
69 // ostream operator
70 ostream& operator << (ostream& osObject, const Time& time1) //creating an instance of a date class, i.e. date1
71 {
72     osObject << time1.sec
73              << "/" << time1.min
74              << "/" << time1.hour;
75     return osObject;
76 }
77
78 // istream operator
79 istream& operator >> (istream& isObject, Time& time1)
80 {
81     isObject >> time1.sec >> time1.min >> time1.hour;
82     return isObject;
83 }
84
85 // Overloading "equal to" operator
86 bool Time::operator==(const Time& otherTime) const
```

```
 87 {
 88     if (sec == otherTime.sec && min == otherTime.min
 89         && hour == otherTime.hour)
 90         return true;
 91     else
 92         return false;
 93 }
 94
 95 // Overloading "not equal to" operator
 96 bool Time::operator!=(const Time& otherTime) const
 97 {
 98     return !(*this == otherTime);
 99 }
100
101 // Overloading "less than or equal to" operator
102 bool Time::operator<=(const Time& otherTime) const
103 {
104     return (*this < otherTime || *this == otherTime);
105 }
106
107 // Overloading "less than" operator
108 bool Time::operator<(const Time& otherTime) const
109 {
110     if ((hour < otherTime.hour) ||
111         (hour == otherTime.hour && min < otherTime.min) ||
112         (hour == otherTime.hour && min == otherTime.min && sec < otherTime.sec))
113         return true;
114     else
115         return false;
116 }
117
118 // Overloading "greater than or equal to" operator
119 bool Time::operator>=(const Time& otherTime) const
120 {
121     return !(*this < otherTime);
122 }
123
124 // Overloading "greater than" operator
125 bool Time::operator>(const Time& otherTime) const
126 {
127     return !(*this <= otherTime);
128 }
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\AirQuality.h

```
1 //
2 // Created by hycwy on 4/10/2022.
3 //
4 #ifndef FINAL_PROJECT_AIR_QUALITY_H
5 #define FINAL_PROJECT_AIR_QUALITY_H
6
7 #include "Date.h"
8 #include "Time.h"
9
10
11 class AirQuality {
12 public:
13     AirQuality();
14
15     AirQuality(Date dateInput, Time timeInput, double t, double r, double a);
16
17     AirQuality(double t, double r, double a);
18
19     void setDate(Date d);
20     void setTime(Time t);
21     void setTemp(double t);
22     void setRelativeHumidity(double r);
23     void setAbsHumidity(double a);
24
25     Date getDate();
26     Time getTime();
27     double getTemp();
28     double getRelativeHumidity();
29     double getAbsHumidity();
30
31 private:
32
33     // data members
34     Date date = Date();
35     Time time = Time();
36     double temp;
37     double relativeHumidity;
38     double absHumidity;
39 };
40 #endif
41
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\dateTest.cpp

```cpp
#include <iostream>
#include <fstream>
#include <iomanip>
#include "vector"
#include "string"
#include "Date.h"
#include "Time.h"
#include "AirQuality.h"


int main() {

    Date date = Date(1,1,2022);
    int day = date.getDay();
    //Expected to be 1, actually get 1
    cout << day << endl;

    int month = date.getMonth();
    //Expected to be 1, actually get 1
    cout << month << endl;

    int year = date.getYear();
    //Expected to be 2022, actually get 2022
    cout << year << endl;

    date.setDay(10);
    date.setMonth(10);
    date.setYear(2023);

    day = date.getDay();
    //Expected to be 10, actually get 10
    cout << day << endl;

    month = date.getMonth();
    //Expected to be 10, actually get 10
    cout << month << endl;

    year = date.getYear();
    //Expected to be 2023, actually get 2023
    cout << year << endl;

    date.setDmy(12,12,2020);
    //Expected to be 12/12/2020, actually get 12/12/2020
```

```
44      date.print();
45      cout << endl;
46
47      Date date1 = Date(21,12,2022);
48      Date date2 = Date(20,12,2022);
49
50      bool checker1 = date1 > date2;
51      cout << boolalpha;
52      //Expected to be true, actually get true
53      cout << checker1 << endl;
54
55      Date date3 = Date(22,1,2022);
56      bool checker6 = date1 > date3;
57      cout << boolalpha;
58      //Expected to be true, actually get true
59      cout << checker6 << endl;
60
61      Date date4 = Date(22,12,2021);
62
63      bool checker7 = date1 > date4;
64      cout << boolalpha;
65      //Expected to be true, actually get true
66      cout << checker7 << endl;
67
68      bool checker2 = date1 >= date1;
69      cout << boolalpha;
70      //Expected to be true, actually get true
71      cout << checker2 << endl;
72
73      bool checker3 = date1 <= date1;
74      cout << boolalpha;
75      //Expected to be true, actually get true
76      cout << checker3 << endl;
77
78      bool checker4 = date2 < date1;
79      cout << boolalpha;
80      //Expected to be true, actually get true
81      cout << checker4 << endl;
82
83      bool checker5 = date1 == date1;
84      cout << boolalpha;
85      //Expected to be true, actually get true
86      cout << checker5 << endl;
```

```
87
88      bool checker8 = date1 != date3;
89      cout << boolalpha;
90      //Expected to be true, actually get true
91      cout << checker8 << endl;
92
93      return 0;
94  }
95
96
97
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\timeTest.cpp

```cpp

#include <iostream>
#include <fstream>
#include <iomanip>
#include "vector"
#include "string"
#include "Date.h"
#include "Time.h"
#include "AirQuality.h"


int main() {

    Time time = Time(1,1,3);

    int sec = time.getSec();
    //Expected to be 1, actually get 1
    cout << sec << endl;

    int min = time.getMin();
    //Expected to be 1, actually get 1
    cout << min << endl;

    int hour = time.getHour();
    //Expected to be 3, actually get 3
    cout << hour << endl;

    time.setSec(10);
    time.setMin(10);
    time.setHour(2);

    sec = time.getSec();
    //Expected to be 10, actually get 10
    cout << sec << endl;

    min = time.getMin();
    //Expected to be 10, actually get 10
    cout << min << endl;

    hour = time.getHour();
    //Expected to be 2, actually get 2
    cout << hour << endl;

```

```
44      time.setSmh(12,12,2);
45      //Expected to be 12:12:2, actually get 12:12:2
46      time.print();
47      cout << endl;
48
49      Time time1 = Time(10,10,10);
50      Time time2 = Time(9,10,10);
51
52      bool checker1 = time1 > time2;
53      cout << boolalpha;
54      //Expected to be true, actually get true
55      cout << checker1 << endl;
56
57      bool checker2 = time1 >= time2;
58      cout << boolalpha;
59      //Expected to be true, actually get true
60      cout << checker2 << endl;
61
62      Time time3 = Time(10,9,10);
63      bool checker3 = time1 > time3;
64      cout << boolalpha;
65      //Expected to be true, actually get true
66      cout << checker3 << endl;
67
68      Time time4 = Time(10,9,9);
69      bool checker4 = time1 > time4;
70      cout << boolalpha;
71      //Expected to be true, actually get true
72      cout << checker4 << endl;
73
74      bool checker5 = time1 != time3;
75      cout << boolalpha;
76      //Expected to be true, actually get true
77      cout << checker5 << endl;
78
79      bool checker6 = time1 == time1;
80      cout << boolalpha;
81      //Expected to be true, actually get true
82      cout << checker6 << endl;
83
84      Time time7 = Time(12,10,10);
85      bool checker7 = time1 < time7;
86      cout << boolalpha;
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\timeTest.cpp

```
 87     //Expected to be true, actually get true
 88     cout << checker7 << endl;
 89
 90     Time time8 = Time(1,12,10);
 91     bool checker8 = time1 < time8;
 92     cout << boolalpha;
 93     //Expected to be true, actually get true
 94     cout << checker8 << endl;
 95
 96     Time time9 = Time(12,10,11);
 97     bool checker9 = time1 < time9;
 98     cout << boolalpha;
 99     //Expected to be true, actually get true
100     cout << checker9 << endl;
101
102     bool checker10 = time7 <= time7;
103     cout << boolalpha;
104     //Expected to be true, actually get true
105     cout << checker10 << endl;
106
107     return 0;
108 }
109
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\AirQuality.cpp

```cpp
//
// Created by hycwy on 4/10/2022.
//

#include "AirQuality.h"

AirQuality::AirQuality() {
    temp = 0.0;
    relativeHumidity = 0.0;
    absHumidity = 0.0;
}

AirQuality::AirQuality(double t, double r, double a) {
    temp = t;
    relativeHumidity = r;
    absHumidity = a;
}

AirQuality::AirQuality(Date dateInput, Time timeInput, double t, double r, double a) {
    date.setDmy(dateInput.getDay(),dateInput.getMonth(),dateInput.getYear());
    time.setSmh(timeInput.getSec(), timeInput.getMin(), timeInput.getHour());
    temp = t;
    relativeHumidity = r;
    absHumidity = a;
}

void AirQuality::setDate(Date d) {
    date.setDmy(d.getDay(), d.getMonth(), d.getYear());
}

void AirQuality::setTime(Time t) {
    time.setSmh(t.getSec(), t.getMin(), t.getHour());
}

void AirQuality::setTemp(double t) {
    temp = t;
}

void AirQuality::setRelativeHumidity(double r) {
    relativeHumidity = r;
}

void AirQuality::setAbsHumidity(double a) {
```

```
44      absHumidity = a;
45  }
46
47  Date AirQuality::getDate() {
48      return date;
49  }
50
51  Time AirQuality::getTime() {
52      return time;
53  }
54
55  double AirQuality::getAbsHumidity() {
56      return absHumidity;
57  }
58
59  double AirQuality::getRelativeHumidity() {
60      return relativeHumidity;
61  }
62
63  double AirQuality::getTemp() {
64      return temp;
65  }
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\CMakeLists.txt

```
cmake_minimum_required(VERSION 3.21)
project(final_test)

set(CMAKE_CXX_STANDARD 14)

set(Headers AirQuality.h Date.h Time.h)

add_executable(AirQualityTest AirQualityTest.cpp Time.cpp Date.cpp AirQuality.cpp)
add_executable(main main.cpp Time.cpp Date.cpp AirQuality.cpp)
add_executable(dateTest dateTest.cpp Date.cpp Time.cpp AirQuality.cpp)
add_executable(timeTest timeTest.cpp Date.cpp Time.cpp AirQuality.cpp)
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\AirQualityTest.cpp

```cpp
#include <iostream>
#include "AirQuality.h"

int main()
{
    // No.1 Default constructor test
    AirQuality a;
    cout << "Actual:"
         << a.getTemp() << " "
         << a.getRelativeHumidity() << " "
         << a.getAbsHumidity() << endl;
    cout << "Expect:0 0 0" << endl;

    // No.2 Constructor with temp, relativeHumidity, absHumidity
    AirQuality b(12.2, 40.1, 50.3);
    cout << "Actual:"
         << b.getTemp() << " "
         << b.getRelativeHumidity() << " "
         << b.getAbsHumidity() << endl;
    cout << "Expect:12.2 40.1 50.3" << endl;

    // No.3 Constructor with all parameters.
    Date d;
    Time t;
    AirQuality c(d, t, 1.1, 2.2, 3.3);
    cout << "Actual:"
         << c.getDate() << " "
         << c.getTime() << " "
         << c.getTemp() << " "
         << c.getRelativeHumidity() << " "
         << c.getAbsHumidity() << endl;
    cout << "Expect:1/1/2000 1/1/1 1.1 2.2 3.3" << endl;

    // No.4 setDate
    Date d2(15, 3, 2022);
    AirQuality a1;
    cout << "Actual:"
         << a1.getDate() << endl;
    cout << "Expect:1/1/2000" << endl;
    a1.setDate(d2);
    cout << "Actual:"
         << a1.getDate() << endl;
    cout << "Expect:15/3/2022" << endl;
```

File - C:\Users\nicho\Desktop\GitHub\CS5008_LastAssignment\AirQualityProcessor\final_test\AirQualityTest.cpp

```
44
45      // No.5 setTime
46      Time t2(40,20,15);
47      AirQuality a2;
48      cout << "Actual:"
49           << a2.getTime() << endl;
50      cout << "Expect:1/1/1" << endl;
51      a2.setTime(t2);
52      cout << "Actual:"
53           << a2.getTime() << endl;
54      cout << "Expect:40/20/15" << endl;
55
56      // No.6 setTemp
57      AirQuality a3;
58      a3.setTemp(15);
59      cout << "Actual:" << a3.getTemp() << endl;
60      cout << "Expect:15" << endl;
61
62      // No.7 setRelativeHumidity;
63      AirQuality a4;
64      a4.setRelativeHumidity(20);
65      cout << "Actual:" << a4.getRelativeHumidity() << endl;
66      cout << "Expect:20" << endl;
67
68      // No.8 setAbsHumidity
69      AirQuality a5;
70      a5.setAbsHumidity(35);
71      cout << "Actual:" << a5.getAbsHumidity() << endl;
72      cout << "Expect:35" << endl;
73
74      // No.9 getAbsHumidity
75      cout << "Actual:" << a5.getAbsHumidity() << endl;
76      cout << "Expect:35" << endl;
77
78      // No.10 getRelativeHumidity
79      cout << "Actual:" << a4.getRelativeHumidity() << endl;
80      cout << "Expect:20" << endl;
81
82      // No.11 getTemp
83      cout << "Actual:" << a3.getTemp() << endl;
84      cout << "Expect:15" << endl;
85  }
86
```