C++ Exercise 2: Find Minimum and Maximum for Pairs

Deadline: Friday, September 11, 2020 by 23:59 EEST Timezone.
How to submit: Check your code into your GitHub repository (`workspace-cpp` or similar) and create a branch `ex02-find-extremes` there.
Grading: This exercise is worth 20‰(or 2%) of the total grade.

Consider a school having one or more students. For every student we know his/her age (a positive integer) and also height in centimeters (a positive double precision number). Every morning the students arrange themselves in a line by seniority – the first one is the youngest one called the *minimum*. S/he is followed by the next youngest, etc. The last student, who is the oldest one is called the *maximum*. In case, if the ages of two students are equal, we compare their height (the shortest one comes first).
You do not need to arrange (sort) the students; only find the minimum and the maximum, and output their respective age and height.

Develop a software that receives an input file containing the total count of the students N on the first line. It is followed by exactly N pairs of numbers: a_i and h_i ($i \in 1, \ldots, N$).
The software should read all the pairs in an array and output the data about the students at both ends of their arranged line (the minimum student and the maximum student).

Input
The first line has a single number N. After that there are exactly N lines; every line contains one positive integer and another positive double; they are space-separated.

Output
The output should contain data for two "extreme" students only on two separate lines.
The first student is the minimal student; the second student is the maximum (sorted by age; or by age+height, if many ages are equal). Heights should be rounded to 5 decimal places in the output.

Limitations

- The number of students N is between 1 and 10000.
- Ages are strictly positive integers; they do not exceed $2 \cdot 10^9$. Heights are strictly positive double numbers.

Sample input **test01in.txt**:

```
3
16 185.3
12 150.0999999
16 180.1
```

Expected output **test01expected.txt**:

```
12 150.10000
16 185.30000
```

Sample input **test02in.txt**:

```
1
2000000000 1.5
```

Expected output **test02expected.txt**:

```
2000000000 1.50000
2000000000 1.50000
```

Sample input **test03in.txt**:

```
6
3 1
3 1.2
3 1.4
5 2.5
5 2.7
5 2.999999
```

Expected output **test03expected.txt**:

```
3 1.00000
5 3.00000
```

Implementation Details

- Your code should contain 3 source files: `Student.cpp`, its header file: `Student.h`, and the class with main function that processes the I/O `ExtremesMain.cpp`.
- All classes and functions should be in the namespace `ds_course`.
- The class `Student` should support the following members (two data attributes and one method to compare it with other students).

```
Student();           // an empty constructor
int age;
double height;
int compareTo(Student ss);
```

- The file `ExtremesMain.cpp` should contain two functions to find the minimum and the maximum student in an array of students:

```
// return the smallest student object in the array ss[count]
Student getMin(Student* ss, int count);
// return the largest student object in the array ss[count]
Student getMax(Student* ss, int count);
```

- Function `compareTo(...)` returns 0, if "this student" equals the "argument student" (both age and height). It returns -1, if "this student" is less than the "argument student". It returns 1, if "this student" is more than the "argument student". For example, the following code should work:

```
Student s0; s0.age = 3; s0.height = 170.2;
Student s1; s1.age = 5; s1.height = 178.2;
Student s2; s2.age = 3; s2.height = 168.2;
cout << s0.compareTo(s0) << endl;  // prints 0
cout << s0.compareTo(s1) << endl;  // prints -1
cout << s0.compareTo(s2) << endl;  // prints 1

Student ss[] = {s0,s1,s2};
cout << getMin(ss,3).height << endl;
// should print 168.2 (the least height of the 2 youngest ones)
```

- Our grading process may also invoke the class `Student` and the functions `getMin(...)` and `getMax(...)` (from `ExtremesMain.cpp`) directly.

Note 1. The last requirement means that you cannot "forget" the input lines (in other contexts it might be very efficient – just scan the input data once and update the maximum and minimum, if necessary). But in your solution you should load the input data into an array and then pass that array to the functions `getMin()` and `getMax()`.

Note 2. What is the format of double precision numbers representing `height`?

Inputs can contain any number readable by "cin" (we will not use floating point notation such as `1.7E2` instead of `170` or `170.0`).

The output should always be rounded to 5 decimal digits after the decimal point. You can achieve this behavior using `iomanip` library in C++. For example,

```
#include <iostream>
#include <iomanip>

int main() {
    double aa[] = {3.14159265358979323, 3, 185.999999};
    for (int i = 0; i < 3; i++) {
        cout << "a[ " << i << "] = " << std::fixed << std::setprecision(5) << a[i] << endl;
    }
}
```

The output is this:

```
a[0] = 3.14159
a[1] = 3.00000
a[2] = 186.00000
```