CPP Problem Design Example

Subject: School Inheritance

Contributor: Jun-Yu Chen

Main testing concept: Class/Inheritance/Overloading/Virtual Functions

Basics	Functions
□ C++ BASICS	□ SEPARATE COMPILATION AND NAMESPACES
□ FLOW OF CONTROL	□ STREAMS AND FILE I/O
□ FUNCTION BASICS	□ RECURSION
□ PARAMETERS AND OVERLOADING	■ INHERITANCE
□ ARRAYS	■ POLYMORPHISM AND VIRTUAL FUNCTIONS
■ STRUCTURES AND CLASSES	□ TEMPLATES
□ CONSTRUCTORS AND OTHER TOOLS	□ LINKED DATA STRUCTURES
■ OPERATOR OVERLOADING, FRIENDS, AND REFERENCES	□ EXCEPTION HANDLING
□ STRINGS	□ STANDARD TEMPLATE LIBRARY
□ POINTERS AND DYNAMIC ARRAYS	□ PATTERNS AND UML

Description：

The governing of public and private schools is considered different as the two types of schools have different education goals while having different funding and student structures. For private schools the main income comes from students' paying high tuition fees comparing to public schools whose main income relies on funding from the government, as schools attracting more students with their good performance should receive more resources. However, even being almost totally different they are still closely related and could not be managed separately, thus an intermediate between the two systems, regarding student transfer and the increase or decrease of admissions of next year is created.

For this task, you have to write a program that creates three types of school (one from each class: `School`, `PrivateSchool`, and `PublicSchool`), transfers students from each to another and calculate the available admissions for the next year given governing rules.

Create a base class called **`School`** that has member variables and functions:

- the **`name`** of the school (a string)
- the **`studentNumber`** of the school currently
- **`studentNumberNextYear`** indicating the number of students the school could have next year, where when constructing is same as the student number this year.
- ***`admissions(float number):`*** adds the number (passed as a parameter) to the total student number this year (if the number is nonnegative),
- ***`dropouts(float number):`*** subtracts the number from the number of students this year (if the number is nonnegative and less than or equal to the student number)
- ***`transfer(float number, School &toSchool):`*** deducts from the student number of current and transfers them to another school (passed as a parameter), implemented calling ***`dropouts(number)`*** and ***`toSchool.admissions(number).`***

Also, create a class called **`PrivateSchool`** that is derived from **`School`**. In a **`PrivateSchool`**, while the first call of ***`dropouts`*** comes without any penalty, starting from the second call, every additional call of ***`dropouts`*** induces a penalty to deduce 100 from its **`studentNumberNextYear`**. Hence, the class must have a data member to keep track of the times of dropouts being called and override the ***`dropouts`*** function.

Finally, create a **`PublicSchool`** class derived from **`School`.** Additionally, please add a member variable, ***`growing_rate`*** **`(=0.05),`** and a member function, ***`apply_growth()`***, which increases number of students able to admit next year by **`studentNumberNextYear`** ***`+= growing_rate`*** * **`studentNumberNextYear`**. **`PublicSchool`** incur penalties when large number of students (>100) leave the school at once. A dropout of such a number induces a loss of

5% of **`studentNumberNextYear`**, truncating the decimal places. Again, the ***`dropouts`*** function must override the one in the base class.

For all 3 classes create constructors (default and with parameters) and overloaded << (output) operator, reuse constructors and operator << of the base class in the derived classes.

Input：

No Inputs.

Output：

Please refer to sample output for output format, consisting of name, studentNumber and studentNumberNextYear. Separated by tab('\t') as written in bold in the next line:

**** ″name\tstudentNumber\tstudentNumberNextYear″**

Error handling：

If any violations stated in the description occurs (Ex. Subtracting more than existing number), we do not do the operation and output "ERROR\n".

Sample Input	Sample Output
According to the given main.cpp in Other Notes	NTUST 12500 12500 NTUT 85000 85000 FJCU 25000 25000 NTUST 12700 12500 NTUST 12500 12500 ERROR NTUST 12500 12500 FJCU 26000 25000 FJCU 25950 25000 FJCU 24950 24900 NTUT 86000 85000 NTUT 86000 89250 NTUT 85000 84787 NTUT 85000 84787 NTUT 84000 80547 NTUST 13500 12500 ERROR FJCU 24950 24900 NTUST 13500 12500

□ **Easy, Only basic programming syntax and structure are required.**
■ **Medium, Multiple programming grammars, and structures are required.**
□ **Hard, Need to use multiple program structures or complex data types.**

Expected solving time:

25 minutes

Other Notes:

```
#include <iostream>
#include "School.h"
#include <string>
```

```
using namespace std;

int main()
{
   //init 3 different account types
   School ntust("NTUST", 12500);
   PublicSchool ntut("NTUT", 85000);
   PrivateSchool fjcu("FJCU", 25000);

   //state info all 3
   cout<<ntust<<endl;
   cout<<ntut<<endl;
   cout<<fjcu<<endl;

   //test all methods on School
   ntust.admissions(200);
   cout<<ntust<<endl;

   ntust.dropouts(200);
   cout<<ntust<<endl;

   ntust.dropouts(100000);
   cout<<ntust<<endl;

   //test all methods on PrivateSchool
   fjcu.admissions(1000);
   cout<<fjcu<<endl;

   fjcu.dropouts(50);
   cout<<fjcu<<endl;

   fjcu.dropouts(1000);
   cout<<fjcu<<endl;

   //test all methods on PublicSchool
   ntut.admissions(1000);
   cout<<ntut<<endl;

   ntut.apply_growth();
   cout<< ntut <<endl;

   ntut.dropouts(1000);
   cout<< ntut <<endl;

   //Transfer method
   cout << ntut << endl;
   ntut.transfer(1000, ntust);
   cout << ntut << endl;
   cout << ntust << endl;

   fjcu.transfer(30000, ntust);
   cout << fjcu << endl;
   cout << ntust << endl;

   return 0;
```

```
}
```