Creative Software Design, Assignment 10-1

Deadline: 2024-11-05 23:59 (No score for late submission)

- Submit your homework by uploading your zip file to the LMS assignment section. Below is an example.

```
13178_Assignment1-1_2024123456.zip
├─ 1.cc
├─ 2.cc
├─ 3.cc
└─ ...
```

- Your zip file name should follow this format:
  13178_Assignment[Assignment-number]_[Student-ID].zip
  - Ex. 13178_Assignment1-1_2024123456.zip
- Source files should be named as **<filename>.cc** *or* **<filename>.cpp**
- **You must submit your solution in the zip file before the deadline.**

1. **Implement the four fundamental arithmetic operations between MyInt class and MyDouble class.**

   A. **Define the `MyInt` class as follows:**

      1. Implement all pre-defined functions.

```
class MyInt {
    int num;

public:
    MyInt(int num);         // Constructor with initializer list

    int getNum() const;     // Returns the integer value of num
    void show() const;      // Displays the value of num
};
```

   B. **Define the `MyDouble` class as follows:**

      1. Implement all pre-defined functions.

```
class MyDouble {
    double num;

public:
    MyDouble(double num);   // Constructor with initializer list

    double getNum() const;  // Returns the double value of num
    void show() const;      // Displays the value of num
};
```

   C. **Define and implement all fundamental arithmetic operations by overloading.**

      1. The calculation result of `MyInt` and `MyInt` should be `MyInt`, and the result of other calculations should be `MyDouble`.
      2. Exceptions for operations such as `5/0` are not considered.
      3. Additional functions or variables can be added if necessary.
      4. Overload the operators `+`, `-`, `*`, `/`.

   D. **Implement the main `main()` function as follows:**

```
int main() {
    MyInt intNum_1(10), intNum_2(20);
    MyDouble doubleNum_1(100.0), doubleNum_2(70.0);

    // Write your code here
}
```

E. Example output of your program (Bold text indicates user input):

```
200
60
120
1.42857
```

F. Submission file: one C++ source file (File name: 1.**cc** or 1.**cpp**)

2. Implement a program to obtain the total score of test problems.

A. Define the `TestProblem` class as follows:

1. **Implement all pre-defined functions.**
2. The initial value of `totalScore` is set to 0.
3. The **constructor** initializes `problemScore` to `0` and inserts "`subScore1`" and "`subScore2`" with initial values of `0` into `subScore`.
4. The **copy constructor** initializes a new `TestProblem` instance by copying another instance.
5. `setSubScore(string key, int _score)`: Sets the sub-score for a specified key.
   i. This method returns void.
   ii. If `_score` is within the range $0 \leq$ `_score` ≤ 50, it updates the sub-score. Otherwise, it displays "`Error`".
6. `getProblemScore()`: Returns the total score of the problem.
7. `showTotalScore()`: A static function that displays the total score of all problems.

```
class TestProblem {
private:
    map<string, int> subScore;
    int problemScore;
    static int totalScore;

public:
    TestProblem();
    TestProblem(const TestProblem& testProblem);
    void setSubScore(string key, int _score);
    int getProblemScore() const;
    void show() const;
    static void showTotalScore();
};
```

B. **Example of the `main()` function:**

```
int main() {
    TestProblem problem1;
    TestProblem problem2 = problem1;

    problem1.setSubScore("subScore1", 30);
    problem1.setSubScore("subScore2", 40);
    problem2.setSubScore("subScore1", 20);
    problem2.setSubScore("subScore2", 40);

    problem1.show();
    cout << "problem1 score: " << problem1.getProblemScore() << endl;
    problem2.show();
    cout << "problem2 score: " << problem2.getProblemScore() << endl;
    TestProblem::showTotalScore();

    return 0;
}
```

C. Example output of your program (Bold text indicates user input):

```
subScore1: 30
subScore2: 40
problem1 score: 70
subScore1: 20
subScore2: 40
problem2 score: 60
total score: 130
```

D. Submission file: one C++ source file (File name: 2.**cc** or 2.**cpp**)

3. Implement functions that exchange specifications between two hardware objects.

A. Define the `HardwareA` and `HardwareB` classs as follows:

1. The data type of `specification` is `int`, and it stores the hardware specification as a private member of both `HardwareA` and `HardwareB` classes.
2. The `HardwareA` and `HardwareB` classes should implement the following:
   i. A constructor to initialize `specification`.
   ii. `setSpecification(int)`: Sets the hardware specification.
   iii. `show()`: Displays the specification.
   iv. `exchange()`: Exchanges the specification with the other hardware object.
3. If `exchange(HardwareA, HardwareB)` is called, it changes the specifications of the two hardware objects using a friend function.
   i. This function is an overloaded exchange function with a return type of void.
4. If `show(HardwareA, HardwareB)` is called, it displays the specifications of both hardware objects using a friend function.
   i. This function has a return type of void.

B. **Example of the `main()` function:**

```
int main() {
    HardwareA hardwareA(10);
    HardwareB hardwareB(20);

    show(hardwareA, hardwareB);
    exchange(hardwareA, hardwareB);
    show(hardwareA, hardwareB);
    exchange(&hardwareA, &hardwareB);
    show(hardwareA, hardwareB);

    return 0;
}
```

C. Example output of your program (Bold text indicates user input):

```
HardwareA, HardwareB: 10, 20
HardwareA, HardwareB: 20, 10
HardwareA, HardwareB: 10, 20
```

D. Submission file: one C++ source file (File name: 3.**cc** or 3.**cpp**)