Creative Software Design, Assignment 4-2

Deadline: 2024-10-01 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. Write a C++ program to register book information. The `Library` class should register the book title, author, and price. After deciding how many books to register, enter the information for each book and display the information of the entered books.

   A. Define a `Library` class in C++ with the following structure:

      1. **Private members:**
         - `string* _title`: Stores the titles of the books.
         - `string* _author`: Stores the authors of the books.
         - `int* _price`: Stores the prices of the books.
      2. **Public members:**
         - `Library(int num)`: A constructor that dynamically allocates memory for `_title`, `_author`, and `_price` by the number of books (`num`).
         - `void setBookInfo(int index, string title, string author, int price)`: A setter function that takes the book title, author, and price, and assigns them to the corresponding index in the arrays.
         - `void showBookList()`: A function to display all the registered book information.
         - `~Library()`: A destructor that deletes the dynamically allocated book title, author, and price.

   B. Additional Instructions

      1. Write the variable and function names exactly as shown in the problem.
      2. Do not use the CString function.
      3. Do not use `malloc()` or `free()`.

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

```
How many register book: 2⏎
Book title: Introduction to software testing⏎
Book author: PUAL AMMAMN AND JEFF OFFUTT⏎
Book price: 45000⏎
Book title: C++ Primer⏎
Book author: Stanley B⏎
Book price: 67000⏎
Book 1: Introduction to software testing
PUAL AMMAMN AND JEFF OFFUTT / 45000
BOOK 2: C++ Primer
Stanley B / 67000
```

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

2. Write a C++ program to sum two times.

- The times are entered through the `main` function.
- Use the `this` pointer to store the hour, minute, and second values in the `Time` class through a `setTime` function.
- Display the sum of the two times using the `showTime()` function in the Time class.

A. Define a `Time` class in C++ with the following structure:

1. **Private Members:**
   - `int _hour, int _minute, int _second`: These are data members for storing hours, minutes, and seconds, all of type integer.
2. **Public Members:**
   - `Time()`: A constructor to initialize `_hour`, `_minute`, and `_second` to zero.
   - `void setTime(int hour, int minute, int second)`:A member function that takes parameters for hour, minute, and second and assigns these values to the private members `_hour`, `_minute`, and `_second`.
   - `void addTime(Time T1, Time T2)`: A member function that sums two `Time` objects. This function uses the `this` pointer to update the current object with the sum of the times.

     ☙ Input parameters:

     i. `Time T1`: First `Time` object.

     ii. `Time T2`: Second `Time` object.
   - `void showTime()`: A member function to display the time values.

B. main function in C++ with the following structure:

1. Variables:
   - `Time T1`: Input the first time.
   - `Time T2`: Input the second time.
   - `Time T3`: Store the sum of `T1` and `T2` using the `addTime` function.
2. Requirements:
   - The total time should be displayed in the format `HH:MM:SS`.
   - The total time can exceed 24 hours.
   - Inputs for hour, minute, and second should be taken from the user.
     - Ensure that minutes and seconds follow the rule: 0 ≤ `minutes`, `seconds` < 60.

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

```
Hours: 12⏎
Minutes: 04⏎
Seconds: 48⏎

Hours: 23⏎
Minutes: 55⏎
Seconds: 52⏎

Total time: 36:00:40
```

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

3. Write a C++ program that finds the sum and difference between two distances.

   - Get the distances from the `main` function.
   - Use the `this` pointer to store distance values (in meters) in the `Distance` class through a setter function (`setDis`).
   - Use the `addDis` and `subDis` functions to find the sum and the absolute difference between the two distances.
   - The result is displayed in the `main` function.

A. Define a `Distance` class in C++ with the following structure:

   1. **Private Members:**
      - `double _meter`: A data member that stores the distance in meters (of type double).
   2. **Public Members:**
      - `Distance()`: A constructor to initialize `_meter` to 0.0.
      - `void setDis(double meter)`: A member function that takes the meter value as a parameter and assigns it to the private member `_meter`.
      - `double getDis()`: A member function that returns the value of `_meter`.

B. Define the `main()` function as follows:

   - Variables:
      - `Distance d1`: Input the first distance.
      - `Distance d2`: Input the second distance.

C. `addDis` and `subDis` functions in C++ with the following details:

1. `double addDis(double dis_1, double dis_2)`
   - A function that sums two distances.
   - **Input parameters:**
     - `double dis_1`: Receives the distance (in meters) from the first `Distance` object.
     - `double dis_2`: Receives the distance (in meters) from the second `Distance` object.
   - **Return:**
     - The sum of `dis_1` and `dis_2` as a double.
2. `double subDis(double dis_1, double dis_2)`
   - A function that calculates the absolute difference between two distances.
   - **Input parameters:**
     - `double dis_1`: Receives the distance (in meters) from the first `Distance` object.
     - `double dis_2`: Receives the distance (in meters) from the second `Distance` object.
   - **Return:**
     - The absolute difference between `dis_1` and `dis_2` as a `double`.

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

```
Enter distance1 value of meters: 12.45⏎
Enter distance2 value of meters: 16.55⏎
The sum between the two distances: 29m
The difference between the two distances: 4.1m
```

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