Programming 2 - Laboratories: Task 4

In this task you have to implement class `ZoneClock`, representing time in different time zones. All class declarations are given in header files. Implement all necessary class functionalities.

In class `ZoneClock` fields `hour` and `min` represents number of hours and minutes respectively. In this task time is represented in 12h format (1 <= `hour` <= 12, 0 <= `min` < 60).

Bool field `format` represents format in which time is displayed on the screen. If format is `true` (default value), time is displayed in 12h format (in the same way, as is stored in class, e.g. 12:23am, 07:00pm).

Every third created `ZoneClock` object should posses format set to `false`, which means, that time is displayed in 24 format (number of hours should by converted for the purpose of display, e.g. 12:23, 19:00).

There are two enumerated types decelerated. Enum type `clock_period` represents time periods for 12h format – am or pm. Enum type `time_zone` represents different time zones (UTC, WET = UTC + 0, CET = UTC + 1 and EET = UTC + 2). In class `ZoneClock` field `period` represent current time period, while filed `zone` represents current time zone (by default UTC).

Before you start, download all files. Analyse given code. You are not allowed to make any changes to any class declaration and to the main function, except uncommenting fragments associated with parts of the task.

Part 1 (2 points)

a) Implement class `ZoneClock` constructor:

```
ZoneClock(int hour = 12 , int min = 0 , clock_period period = clock_period::am);
```

creating a single `ZoneClock` element and setting its proper member fields. Remember, **that every third object** of this class is 24h clock (`format` should be **`false`**).

b) Overload operator <<

```
friend std::ostream& operator<<(std::ostream& out, const ZoneClock& c);
```

printing to the screen time information with proper formatting (see example output). We assume that data passed to constructors are valid.

Part 2 (1 point)

Overload operator >>

```
friend std::istream& operator>>(std::istream& in, ZoneClock& c);
```

which asks user to enter hour, minutes and period (entered as an array of characters, accepted values are: “am” or “pm”). This process should be repeated until proper data are not given.

Part 3 (1,5 points)

Implements methods:

```
void add_minute();
void add_minutes(int minutes);
```

First method add one minute to current time. Second method add specified number of minutes (passed as an argument) to current time. Remember that one minute added may cause change of hour and even change of time period (form am to pm or reverse). Try to use the method `add_minute()` implemented first, while implementing second method `add_minutes(int minutes)`.

Part 4 (1,5 points)

Implements methods:

```
void subtract_minute();
void subtract_minutes(int minutes);
```

First method subtract one minute from current time. Second method subtract specified number of minutes (passed as an argument) from current time. Remember that one minute subtracted may cause change of hour and even change of time period (form am to pm or reverse). Try to use the method `subtract_minute()` implemented first, while implementing second method `subtract_minutes(int minutes)`.

Part 5 (1 points)

Change implementation of overload operator << so that time information is printed together with time zone info, e.g.

00:00 UTC + 0

12:00am UTC + 0

12:57am UTC + 1

If current time zone is e.g. CET, we print to the screen UTC + 1 (while CET = UTC + 1).

Part 6 (1 points)

Implements method:

```
void set_time_zone(const time_zone& tz);
```

This method changes actual time zone in object and update time inside object (e.g. changing UTC to CET update 11:54am to 12:54pm, changing EET to WET update 22:54 to 20:54). Changes are made according to difference in time zones.

Example output

```
--------------- Part 1 ---------------
07:34am
07:34pm
00:00
12:00am
11:57pm
12:00
12:12am
01:23pm
21:09
--------------- Part 2 ---------------
Enter hour:13
Enter minutes:23
Enter (am/pm):am
Wrong time!!
Enter hour:7
Enter minutes:12
Enter (am/pm):am
Enter hour:7
Enter minutes:12
Enter (am/pm):pm
Enter hour:7
Enter minutes:12
Enter (am/pm):pm
07:12am
07:12pm
19:12
--------------- Part 3 ---------------
12:00am
12:00pm
12:00am
01:30am

00:00
12:00
00:00
01:30
13:30
--------------- Part 4 ---------------
01:30am
12:00am
12:00pm
12:00am

13:30
01:30
00:00
12:00
00:00
```

--------------- Part 5 ---------------
00:00 UTC + 0
12:00am UTC + 0
11:57pm UTC + 0
21:09 UTC + 0

--------------- Part 6 ---------------
00:00 UTC + 0
12:00am UTC + 0
12:57am UTC + 1
23:09 UTC + 2

02:00 UTC + 2
01:00am UTC + 1
11:57pm UTC + 0
22:09 UTC + 1