Programming Design In-class Practices

Classes

Ling-Chieh Kung

Department of Information Management
National Taiwan University

Problem 1: `Time::isEarlerThan()`

- Let's build a class **`Time`**:

```
class Time
{
private:
   int hour;
   int minute;
   int second;
public:
   Time(int h, int m, int s);
   void print();
};
```

```
Time::Time(int h, int m, int s)
{
  hour = h;
  minute = m;
  second = s;
}

void Time::print()
{
  cout << hour << ":"
       << minute << ":"
       << second;
}
```

Problem 1: `Time::isEarlerThan()`

- Let's build a class **`Time`**:

```
int main()
{
  Time t1(14, 30, 0);
  Time t2(14, 25, 5);

  t1.print();
  cout << "\n";
  t2.print();

  return 0;
}
```

Problem 1: `Time::isEarlerThan()`

- Please implement a member function
  - Return true if and only if the invoking object's time is earlier than that of **`t`**.
- An example program:

```
bool Time::isEarlierThan(Time t);
```

```
int main()
{
  Time t1(14, 30, 0);
  Time t2(14, 25, 5);

  if(t2.isEarlierThan(t1))
    t2.print(); // 14:25:5

  return 0;
}
```

Problem 1: `Time::isEarlerThan()`

- Given three time moments, find the latest one.
- Input:
  - One line with nine integers: h_1, m_1, s_1, h_2, m_2, s_2, h_3, m_3, and s_3. All are valid time values. h_i, m_i, and s_i are the hour, minute, and second of time i.
  - Separated by white spaces.
- Output:
  - $i^* \in \{1,2,3\}$, where time i^* is the index of the latest time.
- Sample input/output:

Input:
```
14 0 0 15 4 13 15 8 2
```

Output:
```
3
```

Input:
```
14 0 0 16 4 13 15 8 2
```

Output:
```
2
```

Problem 2: `Time::printNicely()`

- Please implement a member function
  - Print out 08:09:06 if **`hour`** is 8, **`minute`** is 9, and **`second`** is 6.
  - A **private** member function may help.
- An example program:

```
void Time::printNicely();
```

```
int main()
{
  Time t1(14, 30, 0);
  Time t2(14, 25, 5);

  if(t2.isEarlierThan(t1))
    t2.printNicely(); // 14:25:05

  return 0;
}
```

Problem 2: `Time::printNicely()`

- Given three time moments, print out the latest one nicely.
- Input:
  - One line with nine integers: h_1, m_1, s_1, h_2, m_2, s_2, h_3, m_3, and s_3. All are valid time values. h_i, m_i, and s_i are the hour, minute, and second of time i.
  - Separated by white spaces.
- Output:
  - The nice format of the latest time.
- Sample input/output:

Input:
```
14 0 0 15 4 13 15 8 2
```

Output:
```
15:08:02
```

Input:
```
4 0 0 6 4 13 5 8 2
```

Output:
```
06:04:13
```

Problem 3: `Time::displayFormat`

- Sometimes we want to print out a time string in the 12-hour format.
- Modify your **`Time`** to provide the two display formats.
  - Under the 12-hour format, print out a space and append "AM" or "PM" at the end.
  - The default format is 24 hours.
  - The values are still stored in one single system. Only the display format differs.
  - If we use one format one time string, we should do that for all time strings.
  - A **static** member variable may help.

```
int main()
{
  Time t1(14, 30, 0);
  Time t2(14, 25, 5);

  t1.printNicely(); // 14:30:00
  cout << "\n";

  Time::print12Hour(true);

  t1.printNicely(); // 02:30:00 PM

  return 0;
}
```

Problem 3: `Time::displayFormat`

- Given three time moments, print out the latest one nicely in the given format.
- Input:
  - Line 1: nine integers: h_1, m_1, s_1, h_2, m_2, s_2, h_3, m_3, and s_3. All are valid time values. h_i, m_i, and s_i are the hour, minute, and second of time i. Separated by white spaces.
  - Line 2: 12 or 24 meaning the display format.
- Output:
  - The nice format of the latest time in the given format.

Problem 3: `Time::displayFormat`

- Sample input/output:

Input:
```
14 0 0 15 4 13 15 8 2
12
```

Output:
```
03:08:02 PM
```

Input:
```
4 0 0 6 4 13 5 8 2
24
```

Output:
```
06:04:13
```

Problem 4: constructor and destructor

- Do we need a copy constructor or a destrucotor? Why or why not?

Problem 5: Event

- Let's implement a class **Event** (in a weak way):

```
class Event
{
private:
  char* name;
  Time start;
  Time end;
public:
  Event(char* n, Time s, Time t);
  ~Event();
  void printNicely();
};
```

```
int main()
{
  char n1[] = "PD";
  Event e1(n1, Time(14, 20, 0),
               Time(17, 20, 0));
  e1.printNicely(); // "PD"
                    // Start: 14:20:00
                    // End:   17:20:00

  return 0;
}
```

- Be careful, there is a member variable that is a pointer!
- We need a default constructor for **Time**. Why?

Problem 6: `Event::setName()`

- Let's implement a member function **`setName()`**:

```
void Event::setName(char* n);
```

- Do we need a copy constructor? Why?

```
int main()
{
  char n1[] = "PD";
  Event e1(n1, Time(14, 20, 0),
               Time(17, 20, 0));
  e1.printNicely();

  Event e2(e1); // copy an object
  char n2[] = "Calculus";
  e2.setName(n2);
  e2.printNicely();
  e1.printNicely(); // "Calculus" ?
                    // run-time error?

  return 0;
}
```

Problem 7: `Event` array

- Try to create an array **`schedule`** to store ten **`Event`**.

```
int main()
{
  char n1[] = "PD";
  Event e1(n1, Time(14, 20, 0),
               Time(17, 20, 0));

  Event schedule[10];
  schedule[0] = e1;

  return 0;
}
```

```
int main()
{
  char n1[] = "PD";
  Event e1(n1, Time(14, 20, 0),
               Time(17, 20, 0));

  Event* schedule = new Event[10];
  schedule[0] = e1;
  delete [] schedule;

  return 0;
}
```

- Is there an error? Why? How to fix it?