CSI2372 – Lab4

This lab is about the inheritance mechanism in C++.

1. Here is a class called `StartTime` with the following attributes and methods

```
char d_hour;
char d_minutes;

inline void get(char& _hour, char& _minutes) const;

inline void set(char _hour, char _minutes);

inline StartTime(char _hour, char _minutes);

inline void print() const;
```

Function Inline:

The **inline** keyword is used in C++ and applies to a function. It indicates the compiler that each call to the inline function should be replaced by the body of this function. In order to generate an executable of reasonable size, it is therefore in practice used for "short" functions in terms of number of instructions.

The **inline** keyword has the advantage of speeding up a program if it regularly uses the inline function. It makes it possible to substantially condense the code, in particular for the accessors of a class. A accessor of the class is typically a function of one line code.

The **Inline** allows declaring and implementing functions directly in a header (.h)

Function const:

In C ++, const at the end of the declaration of a function means that the function is not supposed to change the values of the member variables of the class.

This class will be placed in the `starttime.h` file.

2. Define the `CourseActivity` super-class containing a pointer to a `StartTime`, a duration in minutes and a place. These attributes must be private. A public method for changing the start time, a display method, and a full constructor are also required. The signatures of these methods and the type of the attributes are:

```
StartTime* d_start;
double duration;
string location;

CourseActivity(const StartTime* _start, double
              _duration, const std::string _location);

void reschedule(StartTime* _newStart);

void print() const;
```

Define three public subclasses of `CourseActivity`. These classes are: **`Lecture`**, **`Laboratory`** and **`Tutorial`**. Each of these classes should define a method **`void print()`**; and must include a std::string to contain a *topic*, an *assignment*, and an *exercise*. The class definition must be in the `activity.h` file, while the methods will be defined in `activity.cpp`

3. Add the following constructors:

```
Lecture( const StartTime* _start, double
        _duration, const string location, const
        string& _topic );

Lecture( const CourseActivity& _oActivity, const
         string& topic );

Laboratory( const StartTime& _start, double
           _duration, const string location, const
           string& _assignment );

Tutorial( const StartTime& _start, double
         _duration, const string location, const
         string& _exercise );
```

4. Define a *`main()`* function in **`lab4.cpp`**. Create three instances of `StartTime` for 11:30, 13:00 and 16:00. With these, create a **`Lecture`**, a **`Laboratory`** and a **`Tutorial`** with a duration of 90 minutes. Then display each of these activities.

   Create two additional *`Lecture`* instances from *`Tutorial`* and *`Laboratory`* with the constructor **`Lecture`**`(const CourseActivity & _oActivity, const std::string & topic)`; Display these five activities.

   Then change the time of two conflicting lectures for 19:00 and 7:00, respectively. Display activities again.