The objective of this lab is to:

Understand and practice Inheritance, redefining functionality of base class and multiple inheritance.

Instructions!

1. Please follow the dress code before coming to the lab. Keep your student identity cards with you.
2. This is an individual lab, you are strictly **NOT** allowed to discuss your solutions with your fellow colleagues, even not allowed asking how is he/she is doing, it may result in negative marking. You can **ONLY** discuss with your TAs or with me.
1. Strictly follow good coding conventions (commenting, meaningful variable and functions names, properly indented and modular code.
2. Save your work frequently. Make a habit of pressing **CTRL+S** after every line of code you write.
3. Beware of memory leaks and dangling pointers.

Task 01: [15 Marks]

Implement a class named *Novel* that has three data members: *title of Novel*, *author* and *pages*. Novel class should have the following member functions:

1. A *showDetails( )* function to print the details of the novel i.e title and author on console.
2. Three setter functions *setTitle(string)* that sets the title of Novel, a *setAuthor(string)* functions that sets the author of the Novel and *setPages(int)* .
3. Three *const* getter functions *getTitle( ), getPages( )* and *getAuthor( ).*

Now derive a class *Fiction* from Novel class that has two data members: *type* to store the type of fiction (i.e. Science, Mystry, Romance, History, Horror etc.) and *level* (level of imagination created in the fiction story i.e. 1,2,3 etc). The derived class should have following member functions:

1. An overridden function *showDetails( )* function that invokes showDetails of Novel class to print the details of the novel and also prints the type and level of fiction novel on console.
2. Setter functions *setLevel(int)* and *setType(string)* to set the level and type of the fiction novel.
3. Two *const* getter functions *getLevel( )* and *getType( ).*

Derive another class from *Novel*, named *NonFiction* that has one data member *"Inspiration"* to store the actual person, event or place on which the story is based. It should have following member functions:

4. *setData* that invokes *setTitle*, *setPages* and *setAuthor* functions of Novel class and set the inspiration data member.
5. An overridden function *showDetails( )* function that invokes showDetails of Novel class to print the details of the novel and also prints the inspiration of nonfiction novel.

```
class Novel
{
	string title;
	string author;
	int pages;
public:
	Novel();
	Novel(string, string, int);
	void showDetails() const;

	//setter and getter functions for private
	  data memebers.

};
```

```
// class RoomDimension.

class Fiction:public Novel
{
	int level;
	string type;

public:

	// Constructor, setters,
	getters, showDetails and
	other supported functions.

};
```

Write a menu based driver program to show your working.

Task 02: [15 Marks]

The programming task requires you to implement a class named *MobilePhone*. A mobile phone is merely a device for general purposes like voice calling, audio/video recording, messaging, etc. The definition to which is given below:

```
class MobilePhone
{
private:
        int mfgID;
        string mfgDate;
        string mfgName;
        bool state;

public:

        // Both of the constructors display "Phone made by the factory!"
        MobilePhone (); // The default constructor
        MobilePhone (int, string, string); // Parameterized constructor

        // Setters
        void setMfgID (int);
        void setMfgDate (string);
        void setMfgName (string);

        // Getters
        void getMfgID () const;
        void getMfgDate () const;
        void getMfgName () const;


       bool turnOn (); // Sets the state of phone to true also displaying the message
                       // "Phone turned on" and returning the state value

       void makeVoiceCall (int); // The function should accept a phone number as
                                 // parameter and display "Calling to [number]..."

       void recordAudio (); // The function should "start recording" the audio by
                            // displaying the message "Speak into the microphone."

       void sendSMS (string, int); // The function should accept a message as string
                                   // as well as a recipient as int and display the
                                   // message "The message has been sent."

       void turnOff (); // The function displays "Phone turned off" and sets the state
                        // of the phone to false. A mobile can perform no more
                        // functions when it is turned off so you have to handle the
                        // conditions too

       ~MobilePhone (); // Destructor should output "Phone destroyed..."

};
```

1. Inherit a class *ApplePhone* from the *MobilePhone* class using *public* access. This class would have a data member named *iOSVersion*. The class will also have following public functions:

    a) `ApplePhone (); // display message “Hello, Apple!”`
    b) `void launchSiri (); // display message “Talking to Siri”`
    c) `void launchPaper (); // display message “Using Paper app”`
    d) `string getiOSVersion (); // returns iOS Version of phone`

e) `void setiOSVersion (string); // sets iOS Version of phone`
f) `~ApplePhone (); // display message "Apple phone destroyed…"`

Main Function Implementation:
For the main(), you have to write a sequence where the apple's phone is manufactured by the factory, then its turned on, the user makes a voice call, then he reads about stories in Paper app and then he turns off the phone. Call functions one after the other to make this work.

2. Inherit a class *AndroidPhone* from the *MobilePhone* class using *protected* access. This class will have *string* data members namely *kernelVersion* and *androidVersion*. Following are the public member functions to be implemented:

a) `AndroidPhone (); // display message "Hello, Android!"`
b) `void launchSubwaySurfers (); // display message "Playing Subway Surfers"`
c) `void launchFlynx (); // display message "Using Flynx browser"`
d) `string getKernelVersion (); // returns kernel version of phone`
e) `void setiOSVersion (string); // sets kernel version of phone`
f) `string getiOSVersion (); // returns android version of phone`
g) `void setiOSVersion (string); // sets android version of phone`
h) `~AndroidPhone (); // display message "Android phone destroyed…"`

Main Function Implementation:
For the main(), you have to write a sequence whereby, after manufacturing of android phone, the user turns it on, browses web using Flynx, turns it off and then tries to play Subway Surfers. In this case, it should show an error. Call functions one after the other to make this work.

3. Lastly, inherit a class named *WindowsPhone* from the *MobilePhone* class using *private* access specifier. Data members are *productName*, *buildNumber* both of the type *string*. Following member functions require implementation:

a) `WindowsPhone (); // display message "Hello, Windows!"`
b) `void launchFoundbite (); // display message "Checking Foundbite newsfeed"`
c) `void launchXboxGame (); // display message "Playing Xbox game"`
d) `string getProductName (); // returns product name of phone`
e) `void setProductVersion (string); // sets product name of phone`
f) `string getBuildNumber (); // returns build number of phone`
g) `void setBuildNumber (string); // sets build number of phone`
h) `~WindowsPhone (); // display message "Windows phone destroyed…"`

Main Function Implementation:
We have provided the *main( )* code for this one. Stay blessed!

```
int main()
{
	WindowsPhone wp;

	wp.turnOn();
	wp.launchXboxGame();
	wp.sendSMS("Text messaging not allowed in lab!", 02323);
	wp.turnOff();
}
```

Test the code above and get ready for the questions to be asked in evaluation.
It is also compulsory for you to observe the behavior of objects keeping the inheritance concepts in mind.