The objective of this lab is to:

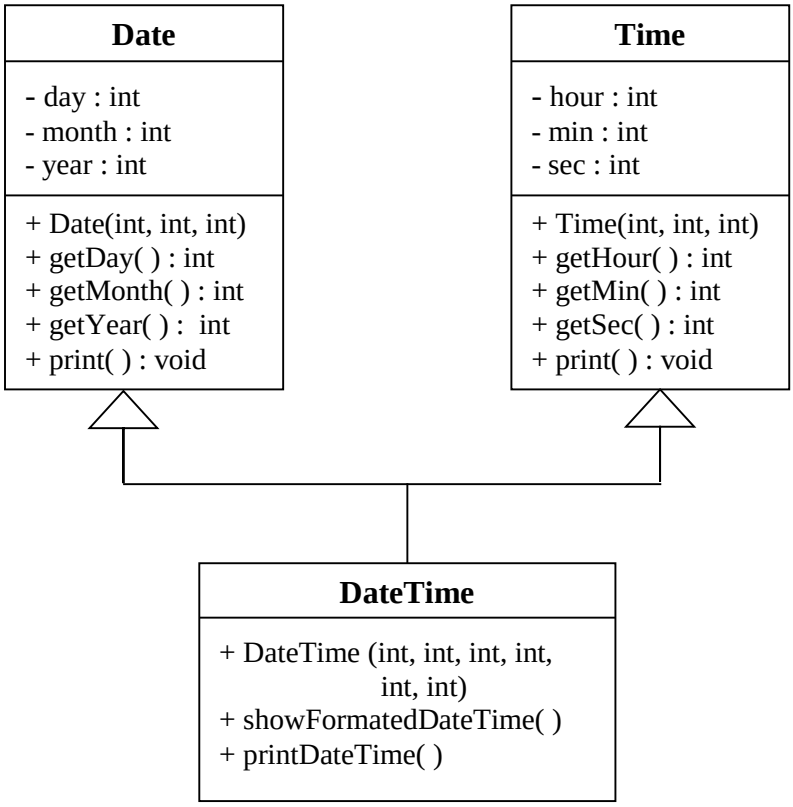
Understand and practice multiple inheritance and UML notations.

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.

Task 01: [10 Marks]

Implement the following Inheritance hierarchy:



Assume the access specifier of all data members is private and functions as public. *print( )* function of each class should print respective data members. printDateTime( ) should print DateTime as:

dd/mm/yyy hh:mm:ss

while *showFormatedDateTime( )* should display date time as **Month day, year hh:mm:ss**. For example May 28, 2015 11:30:15.

Demonstrate your work in *main( )* function.

Task 02: [15 Marks]

Radio Frequency IDentification (RFID) chips are small tags that can be placed on a product. They behave like wireless barcodes and can wirelessly broadcast an identification number to a receiver. One application of RFID chips is to use them to aid in the logistics of shipping freight. Consider a shipping container full of

items. Without RFID chips, a human has to manually inventory all of the items in the container to verify the contents. With an RFID chip attached to the shipping container, the RFID chip can electronically broadcast to a human the exact contents of the shipping container without human intervention.

To model this application, write a base class called ShippingContainer that has a container ID number as an integer. Include member functions to set and access the ID number. Add a virtual function called getManifest that returns an empty string. The purpose of this function is to return the contents of the shipping container.

Create a derived class called ManualShippingContainer that represents the manual method of inventorying the container. In this method, a human simply attaches a textual description of all contents of the container. For example, the description might be "4 crates of apples. 10 crates of pears." Add a new class variable of type string to store the manifest. Add a function called setManifest that sets this string. Override the getManifest function so that it returns this string.

Create a second derived class called RFIDShippingContainer that represents the RFID method of inventorying the container. To simulate what the RFID chips would compute, create an add function to simulate adding an item to the container. The class should store a list of all added items (as a string) and their quantity using the data structures of your choice. For example, if the add function were invoked three times as follows:

```
rfidContainer.add("crate of pears"); // Add one crate of pears
rfidContainer.add("crate of apples"); // Add one crate of apples
rfidContainer.add("crate of pears"); // Add one crate of pears
```

At this point, the data structure should be storing a list of two items: crate of apples and crate of pears. The quantity of apples is 1 and the quantity of pears is 2. Override the getManifest function so that it returns a string of all items that is built by traversing the list of items. In the example above, the return string would be "2 crates of pears. 1 crate of apples."

Finally, write a main program that creates an array of pointers to six ShippingContainer objects. Instantiate the array with three ManualShippingContainer objects and three RFIDShippingContainer objects. For the ManualShippingContainer objects, you will have to invoke setManifest to set the contents. For the RFIDShippingContainer objects, you will have to invoke add to set the contents (although, if this were real, the contents of the container would "add" themselves via the RFID chips instead of requiring a human to type them in). Finally, write a loop that iterates through all ShippingContainer pointers and outputs each object's manifest along with the shipping container ID. You may need to convert an integer into a string. A simple way to do this is: string str = to_string(intVar)

Task 03: [15 Marks]

Consider the following classes:

```
class Employee
{
  public:
    Employee( );
    Employee(string the_name, string the_ssn);
    string get_name() const ;
    string get_ssn( ) const; double get_net_pay( ) const;
    void set_name(string new_name);
    void set_ssn(string new_ssn);
    void set_net_pay(double new_net_pay);
    void print_check() const ;
  private:
```

```
    string name;
    string ssn;
    double net_pay;
};

class SalariedEmployee : public Employee
{
  public:
    SalariedEmployee( );
    SalariedEmployee (string the_name, string the_ssn,double
the_weekly_salary);
    double get_salary( ) const;
    void set_salary(double new_salary);
    void print_check( );
  private:
    double salary; //weekly
};
```

You are required to define a class called Administrator, which is to be derived from the class SalariedEmployee. You are allowed to change the access specifiers of base class (private to protected). You are required to supply the following additional data and function members:

- A member variable of type string that contains the administrator's title (such as Director or Vice President).
- A member variable of type string that contains the company area of responsibility (such as Production, Accounting, or Personnel).
- A member variable of type string that contains the name of this administrator's immediate supervisor.
- A protected member variable of type double that holds the administrator's annual salary. It is possible for you to use the existing salary member if you did the change recommended earlier.
- A member function called set_supervisor, which changes the supervisor name.
- A member function for reading in an administrator's data from the keyboard.
- A member function called print, which outputs the object's data to the screen.
- An overloading of the member function print_check() with appropriate notations on the check.