Object Oriented Programming – Fall 22
(BS-CS-F22)
Lab-10
Lab Instructor: Maa'm Sanam Ahmad

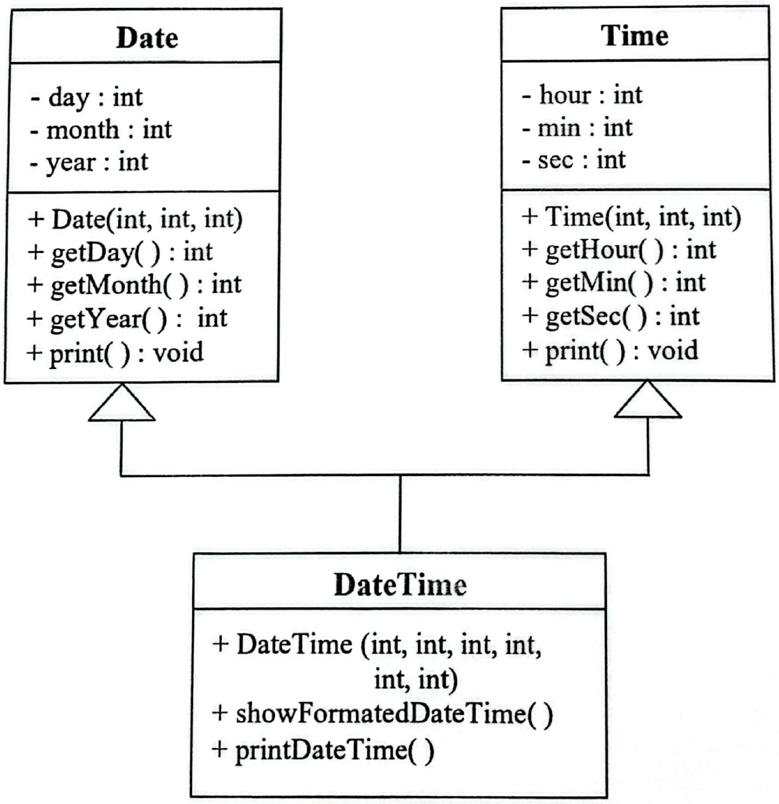
Instructions:

- **Indent your code properly.**
- **Use meaningful variable and function names.**
- **Use the camelCase notation.**
- **Use meaningful prompt lines/labels for all input/output.**
- **Do NOT use any GLOBAL variable(s). However, global named constants may be used.**
- **This is an individual lab, you are strictly NOT allowed to discuss your solution with fellow colleagues, even not allowed to ask how is he/she is doing, it may result in negative marking. You can ONLY discuss with your TAs or with me. • Anyone caught in an act of plagiarism would be awarded an "F" grade in this Lab.**

Do Validations on inputs where required otherwise 1 mark will be deducted for every wrong validation.

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]

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.