CS162 Proficiency Demonstration– Week 10

1. You will be divided into groups to do the demo. Get checked off for your previous lab during this lab. Pay close attention to these **requirements**:
    a. **NO global variables**
    b. **All class interface in .h, class implementation in .cpp, and a driver.cpp with main.**
    c. **All class data members must be private or protected.**
    d. The information in a prototype supplied in the question may not be changed, but you can add
2. **Wait to begin** until informed by the Proctors.
   You will be **given 1 hour** from the time the Proctors begin.
3. You are encouraged to **spend time with design**; scratch paper is provided.
4. Create a directory called test, **mkdir test**
    a. Change into that directory to create and test all your files, **cd test**.
    b. You must stay in this directory
5. Begin entering in your code using a **Linux editor**
    a. You may use vi, vim, or emacs as your editor
6. You are also allowed to **compile, test, and debug** your work.
7. **You will be given 13-15 points for complete work and 0 - 3 points for incomplete work**
8. **When you are finished**, wait for the Proctor to check you off.
    a. Give the Proctor all your design and question material.
    b. Show, compile, and run your program for the proctor.

Class/Main Template/Libraries for Common Built-in Functions:

```
#include <iostream>      /* cin, cout, endl */
#include <cstdlib>       /* srand(), rand(), atoi() */
#include <ctime>         /* time() */
#include <cstring>       /* strlen(), strcmp(), strcpy() */
#include <string>        /* C++ strings: size(), at() */


using namespace std;

//base class interface file, base.h
class base {
   public:
      base();
      ~base();
      void operator=(const base &);
};

//class type implementation, base.cpp
base::base() {

}
base::~base() {

}
void base::operator=(const base &other){

}

//derived class interface file, derived.h
class derived : public base {
   public:
      derived();
};

//class type implementation, derived.cpp
derived::derived() {

}


//driver file to test your class, driver.cpp
int main () {
  srand (time(NULL));  //seed random generator

  return 0;
}
```

Inheritance

Define a **Car** class that stores the car's **id number** as a C++ string and **number of doors** as **protected members**. Add appropriate constructors, accessor functions, and mutator functions. Also define a function named **getMaxSpeed** that returns an integer with the value of zero.

int getMaxSpeed();

Next, define a **RaceCar** class that is **derived from Car**. The RaceCar class should have a **private member** variable named **sponsor** that stores the sponsor of the race car as a C++ string. Add appropriate constructors, mutator functions, and accessor functions for the sponsor variable. Redefine the **getMaxSpeed** function to return 150 if the race car has more than two doors and 200 if the car is less than or equal to two doors.

In the driver,

- Create a Car object and RaceCar pointer.
- Ask the user for the information to set the data members for a Car object.
- Print the max speed of a Car.
- Ask the user for the number of race cars.
- Create a dynamic array of RaceCar objects.
- Ask the user for the information to set the data members for each RaceCar object
- Print the max speed for all the race cars.
- Don't forget to delete your RaceCars!

Big Three

Define a **Doctor** class that has the following **private** data members: doctor's **specialty** (such as "Pediatrician," "Obstetrician," "General Practitioner," etc.) as a C++ string, **number of patients**, and a **dynamic array of Patient** objects. Be sure your class has the appropriate constructors, accessor functions, and mutator functions, as well as the Big Three!!!

Define a **Patient** class that has a private data member for a **name** as a C++ string. Create the appropriate constructors, accessor functions, and mutator functions for the class.

In the driver,

- Write a local function, **void printDocInfo(Doctor d)**, to print the Doctor and Patient information. This will print the Doctor data members and Patient names
- In main…
    - Create a Doctor.
    - Ask the user for the information to set the data members for a Doctor.
    - Create the dynamic array of Patient objects for the Doctor.
    - Ask the user for the information to set all the Patient names.
    - Call printDocInfo() to print the doctor information with patient names.