Lab 7: Mid Preparation Lab

Course Code: CSE 4302
Week: 7
Topics: Basics of Inheritance, Type of Inheritance, and Previous topics
Instructors:
Faisal Hussain, Assistant Professor, CSE
Md. Bakhtiar Hasan, Assistant Professor, CSE
Ishmam Tashdeed, Lecturer, CSE
Farzana Tabassum, Lecturer, CSE

Upload and Turn In your `.cpp` files to **Lab 7** on **Google Classroom**. If you are unable to finish all the tasks, **submit the tasks you have completed within the given time** (You have the chance to complete the remaining ones individually later on and demonstrate them during the next week's lab evaluation). No late submissions are allowed.
Write your code in a single file and name it `XXX_L7.cpp` (Replace `XXX` with the last three digits of your student ID).
After the submission deadline, you will undergo a one-on-one evaluation by one of the course instructors. The instructor will ask you to show your submissions on Google Classroom. They may ask you to run the code locally and ask you questions about it.

Lab Tasks

Put all the tasks in the same file.

1.1 Who is THE Person

The `Person` class represents a generic human entity in an organization. It encapsulates common personal information and provides shared behaviors for all derived roles such as students, faculty members, and visitors.
Note that you may need additional members to achieve the goal of the task. You must convert member functions into `const` member functions (if appropriate). Use appropriate parameters (not mentioned in the instruction)
Class Specification:

- **Member Variables:**
  - `name` (`String`)
  - `age` (`int`)
  - `personID` (`int`) - Ensure that each object is initialized with an ID. The IDs are sequentially generated (automatically). No two objects will have the same ID.
- **Member/Non-member Functions:**

- Zero Argument Constructor - initialize the name "`John Doe`" and age to `0`.
- Argumented Constructor - initialize the members with the given values.
- Destructor - Print "`~Person() is called`".
- `displayDetails()`: Prints the `name`, `age`, and `personID`.
  e.g. "`Faisal, Age: 30, P_ID: 5`"
- Getter and Setter function.
- Operator `==`: Returns `true` if the `name` and the `age` are same.
- Write appropriate functions so that the following code does the same as `displayDetails()`. [Discussion in the lab may be necessary]
  ```
  Person p1, p2;
  cout<<p1<<p2;
  ```

Write the definition of `Person` class following aforementioned requirements.

1.2 You ARE the Student

An object of `Student` class has similar attributes and behaviors as an object of `Person` class. In addition to that, there are few attributes and behaviors.

Class Specification: (Additional Members)

- **Member Variables:**
  - `sid` (`int`)
- **Member Function:**
  - Getter and Setter function.

Write the definition of `Student` class following aforementioned requirements.

1.3 Faculty?

An object of `Faculty` class has similar attributes and behavior as an object of `Person` class. In addition to that, there are few attributes and behaviors.

Class Specification: (Additional Members)

- **Member Variables:**
  - `designation` (`string`)
- **Member Function:**
  - Getter and Setter function.
  - Zero Argument Constructor - Name will be "`Mr. X`" , age is `22`, and designation is `Lecturer`.
  - Argumented Constructor
  - Destructor - Print "`~Faculty() is called`".

Write the definition of `Faculty` class following aforementioned requirements.

1.4 Why Are You Here

An object of `Visitor` class has similar attributes and behavior as an object of `Person` class. In addition to that, there are few attributes and behaviors.

Class Specification: (Additional Members)

- **Member Variables:**
  - `visitingPurpose` (`string`)
- **Member Function:**
  - Getter and Setter function.

Write the definition of `Visitor` class following aforementioned requirements.

*** Use of appropriate type of inheritance, so that members of `Person` class cannot be invoked using `Visitor` object*

Add the following code segment after your class definition.

```
void TestPerson()
{
    Person p1, p2("Faisal",30);

      if(p1.getName()=="John Doe" && p1.getAge() == 0 &&p1.getPersonID()==1) cout<< "Zero Arg Constructor -
Passed"<<endl;
    else  cout<< "Zero Arg Constructor - Failed!!!!"<<endl;

      if(p2.getName()=="Faisal" && p2.getAge() == 30 &&p2.getPersonID()==2) cout<< "Argumented Constructor -
Passed"<<endl;
    else  cout<< "Argumented Constructor - Failed!!!!"<<endl;

    if(p1==p1) cout<< " operator == - Passed"<<endl;
    else cout<< " operator == - Failed!!!!"<<endl;

}

void TestStudent(){
    Student s1;
}
void TestFaculty(){
    Faculty f;
       if(f.getName()=="Mr.  X"&&f.getAge()==22&&f.getDesignation()=="Lecturer")  cout<<  "Zero  Arg  Constructor
Faculty - Passed"<<endl;
    else cout<< "Zero Arg Constructor Faculty - Failed!!!!"<<endl;
}
void TestVisitor()
{
    Visitor v;
}

int main()
{
    TestPerson();
    TestStudent();
    TestFaculty();
    TestVisitor();
}
```