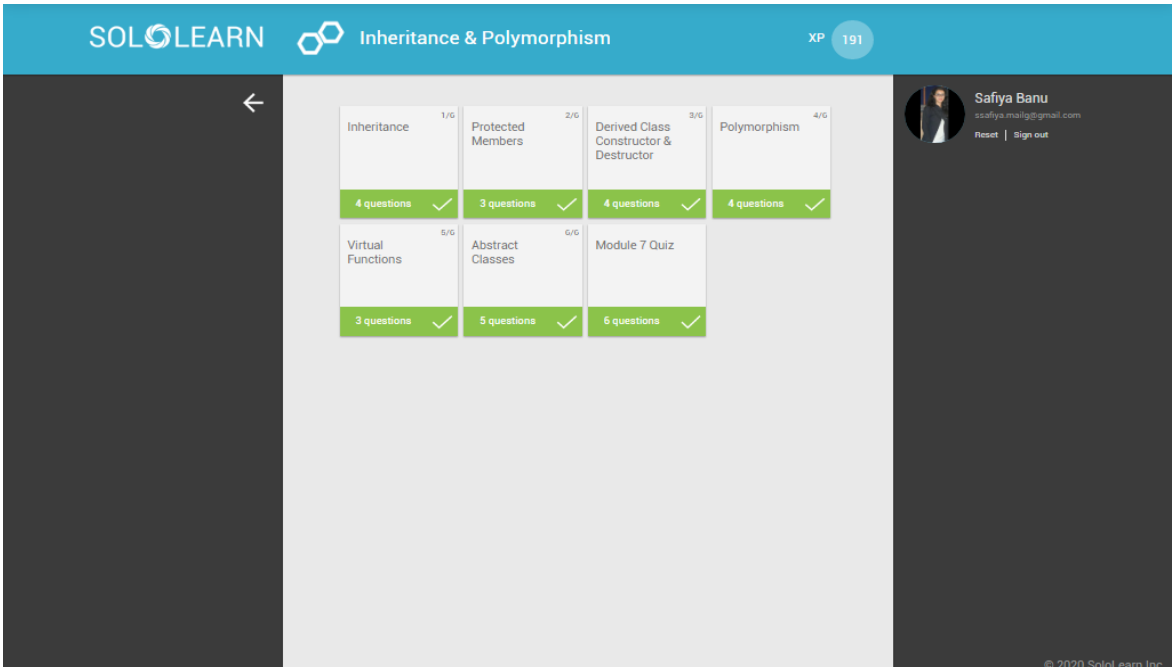
DAILY ASSESSMENT FORMAT

Date:	**26/06/2020**	**Name:**	**SAFIYA BANU**
Course:	**C++**	**USN:**	**4AL16EC061**
Topic:	**Module 7 Inheritance and polymorphism**	**Semester & Section:**	**8TH B**
Github Repository:	**Safiya-Courses**		



Inheritance

Inheritance is one of the most important concepts of object-oriented programming.
Inheritance allows us to define a class based on another class. This facilitates greater ease in creating and maintaining an application.

The class whose properties are inherited by another class is called the Base class. The class which inherits the properties is called the Derived class. For example, the Daughter class (derived) can be inherited from the Mother class (base).
The derived class inherits all feature from the base class, and can have its own additional features.

Access Specifiers

Up to this point, we have worked exclusively with public and private access specifiers.
Public members may be accessed from anywhere outside of the class, while access to private members is limited to their class and friend functions.

Type of Inheritance

Access specifiers are also used to specify the type of inheritance.
Remember, we used public to inherit the Daughter class: `class Daughter: public Mother`
private and protected access specifiers can also be used here.

Public Inheritance: public members of the base class become public members of the derived class and protected members of the base class become protected members of the derived class. A base class's private members are never accessible directly from a derived class, but can be accessed through calls to the public and protected members of the base class.

Protected Inheritance: public and protected members of the base class become protected members of the derived class.

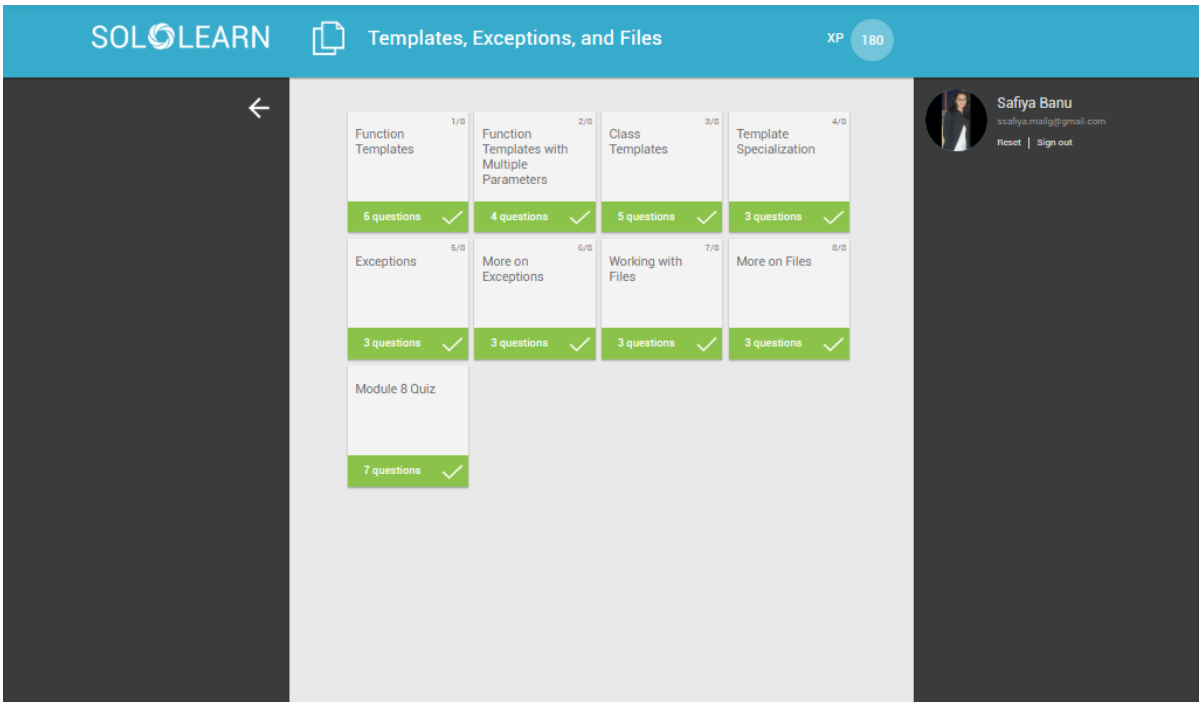
Private Inheritance: public and protected members of the base class become private members of the derived class.

Polymorphism

The word polymorphism means "having many forms".
Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance.

C++ polymorphism means that a call to a member function will cause a different implementation to be executed depending on the type of object that invokes the function.

Date:	**26/06/2020**	**Name:**	**SAFIYA BANU**
Course:	**C++**	**USN:**	**4AL16EC061**
Topic:	**Module 8** **Templates, Exceptions and Files**	**Semester & Section:**	**8TH B**
Github Repository:	**Safiya-Courses**		



Function Templates

Functions and classes help to make programs easier to write, safer, and more maintainable.
However, while functions and classes do have all of those advantages, in certain cases they can also be somewhat limited by C++'s requirement that you specify types for all of your parameters.

Function templates also make it possible to work with multiple generic data types. Define the data types using a comma-separated list.
Let's create a function that compares arguments of varying data types (an int and a double), and prints the smaller one.template <class T, class U>

Class Templates

A specific syntax is required in case you define your member functions outside of your class - for example in a separate source file.
You need to specify the generic type in angle brackets after the class name.

Template Specialization

In case of regular class templates, the way the class handles different data types is the same; the same code runs for all data types.
Template specialization allows for the definition of a different implementation of a template when a specific type is passed as a template argument.

For example, we might need to handle the character data type in a different manner than we do numeric data types.

Exceptions

Problems that occur during program execution are called exceptions.
In C++ exceptions are responses to anomalies that arise while the program is running, such as an attempt to divide by zero.

Catching Exceptions

A try block identifies a block of code that will activate specific exceptions. It's followed by one or more catch blocks. The catch keyword represents a block of code that executes when a particular exception is thrown.
Code that could generate an exception is surrounded with the try/catch block.
You can specify what type of exception you want to catch by the exception declaration that appears in parentheses following the keyword catch.
For example:

```
try {
int motherAge = 29;
int sonAge = 36;
if (sonAge > motherAge) {
```

```
throw 99;
}
}
catch (int x) {
cout<<"Wrong age values - Error "<<x;
}

//Outputs "Wrong age values - Error 99"
```

Working with Files

Another useful C++ feature is the ability to read and write to files. That requires the standard C++ library called fstream.
Three new data types are defined in fstream:
ofstream: Output file stream that creates and writes information to files.
ifstream: Input file stream that reads information from files.
fstream: General file stream, with both ofstream and ifstream capabilities that allow it to create, read, and write information to files.

File Opening Modes

An optional second parameter of the open function defines the mode in which the file is opened. This list shows the supported modes.

Mode Parameter	Meaning
ios::**app**	append to end of file
ios::**ate**	go to end of file on opening
ios::**binary**	file open in binary mode
ios::**in**	open file for reading only
ios::**out**	open file for writing only
ios::**trunc**	delete the contents of the file if it exists