**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **24/06/2020** | **Name:** | **Lepakshi Y V** |
| **Course:** | **C++** | **USN:** | **4AL17EC044** |
| **Topic:** | * **Classes and objects** * **More on classes** | **Semester & Section:** | **6th sem A sec** |
| **Github Repository:** | **Lepakshi-044** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report – Report can be typed or hand written for up to two pages.**  **What is an Object: O**bject **O**riented **P**rogramming is a programming style that is intended to make thinking about programming closer to thinking about the real world. In programming, **objects**are independent units, and each has its own **identity**, just as objects in the real world do? In programming, an object is **self-contained**, with its own **identity**. It is separate from other objects. Each object has its own **attributes**, which describe its current state. Each exhibits its own **behavior**, which demonstrates what they can do.  **What is a Class:** Objects are created using **classes**, which are actually the focal point of OOP? The class **describes**what the object will be, but is separate from the object itself. Each class has a **name**, and describes **attributes**and **behavior**.  **Methods: Method**is another term for a class' behaviour. A method is basically a **function**that belongs to a class.  **Creating a Class:** Let's create a class with one public method, and have it print out "Hi".  The next step is to instantiate an object of our **Bank Account**class, in the same way we define variables of a type, the difference being that our object's type will be **Bank Account**.  **Abstraction:** Data **abstraction**is the concept of providing only essential information to the outside world. It's a process of representing essential features **without including implementation details**. **Abstraction**means, that we can have an idea or a concept that is completely separate from any specific instance.  **Encapsulation:** Part of the meaning of the word **encapsulation**is the idea of "surrounding" an entity, not just to keep what's inside together, but also to **protect**it.  **Constructors:** When creating an object, you now need to pass the constructor's parameter, as you would when calling a function.  **Scope Resolution Operator:** The **double colon** in the source file (.cpp) is called the **scope resolution operator**, and it's used for the constructor definition: #include  **Destructors:** The name of a **destructor**will be exactly the same as the class, only prefixed with a **tilde (~)**. A destructor can't return a value or take any parameters.  **Friend Functions:** Normally, private members of a class cannot be accessed from outside of that class. However, declaring a **non-member**function as a **friend**of a class allows it to access the class' private members. This is accomplished by including a declaration of this external function within the class, and preceding it with the keyword **friend**. In the example below, **someFunc()**, which is not a member function of the class, is a friend of **MyClass**and can access its private members.  **Operator Overloading:** Most of the C++ built-in operators can be redefined or **overloaded**. Thus, operators can be used with user-defined types as well (for example, allowing you to **add**two objects together). |