**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **23/06/2020** | **Name:** | **Kirti B S** |
| **Course:** | **C++ Programming** | **USN:** | **4AL18EC026** |
| **Topic:** | * **Classes and Object** * **More on Classes** | **Semester & Section:** | **4th sem ‘A’ section.** |
| **Github Repository:** | **Kirti BS** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**  **C:\Users\hello\Pictures\Screenshots\Screenshot (294).png**  **C:\Users\Pawan\Desktop\w3.PNG** |
| **REPORT**  **In today’s session I have learnt about:**  **Chapter 1: Classes and Object**   * **What is an Object?**   **Object is a real world entity, for example, chair, car, pen, mobile, laptop etc. In other words, object is an entity that has state and behavior. Here, state means data and behavior means functionality. Object is a runtime entity, it is created at runtime.**   * **What is a Class?**   **A class in C++ is the building block, that leads to Object-Oriented programming. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.**  **Example of Class**   * **Abstraction:**   **Data Abstraction is a process of providing only the essential details to the outside world and hiding the internal details, i.e., representing only the essential details in the program.**   * **Encapsulation:**   **Encapsulation is a process of combining data members and functions in a single unit called class.**  **Example of Encapsulation**   * **Constructors:**   **Constructors are special class members which are called by the compiler every time an object of that class is instantiated. Constructors have the same name as the class and may be defined inside or outside the class definition.**  **Chapter 2: More on Classes**   * **Separate files for Classes:**   **C++ classes (and often function prototypes) are normally split up into two files. The header file has the extension of . h and contains class definitions and functions.**   * **Destructors:**   **Destructor is another special member function that is called by the compiler when the scope of the object ends.**   * **Selection Operators:**   **This syntax is used to select a data member or a member function given a pointer to an object (->) or an object (.) Selecting a data member means getting direct access to a members data, at most objects won't allow that as it breaks the hiding (or encapsulation) of its data.**   * **Const Objects:**   **The const member functions are the functions which are declared as constant in the program. The object called by these functions cannot be modified. It is recommended to use const keyword so that accidental changes to object are avoided. A const member function can be called by any type of object.**   * **Member Initializers:**   **C++ provides member initializers to eliminate such unnecessary default initialization. Again, the entry class has two members of class type, name and value . C++ upholds the initialization guarantee by applying the default constructors to name and value as part of the entry constructor.**   * **The Friend Keyword:**   **A friend function in C++ is a function that is preceded by the keyword “friend”. When the function is declared as a friend, then it can access the private and protected data members of the class.**   * **The This Keyword:**   **In C++ programming, this is a keyword that refers to the current instance of the class. There can be 3 main usage of this keyword in C++. It can be used to pass current object as a parameter to another method. It can be used to refer current class instance variable.**   * **Operator Overloading:**   **Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type. Operator overloading is used to overload or redefines most of the operators available in C++. It is used to perform the operation on the user-defined data type.**   * **Composition : Part 1** * **Composition : Part 2** |