**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **25/06/2020** | **Name:** | **Bhavana.B** |
| **Course:** | **C++** | **USN:** | **4AL18EC009** |
| **Topic:** | **Inheritance and polymorphism** | **Semester & Section:** | **4th sem A section** |
| **Github Repository:** | **Bhavana-b** |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **FORENOON SESSION DETAILS** | | | |
| **Image of session:** | | | |
| **Report:**    **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 function to be executed depending on the type of object that invokes the function. ... This is how polymorphism is generally used.**    **inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. ... In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class.**    **The word polymorphism means having many forms. ... Real life example of polymorphism, a person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism.** | | | |
| **Date:** | **25/06/2020** | **Name:** | **Bhavana.B** | |
| **Course:** | **C++** | **USN:** | **4AL18EC009** | |
| **Topic:** | **Templets , expections and files** | **Semester & Section:** | **4th sem A section** | |
| **Github Repository:** | **Bhavana-b** |  |  | |

|  |
| --- |
| **AFTERNOON SESSION DETAILS** |
| **Image of session:** |
| **Report *:***    **Templates are among the most powerful features of C++, but they remain misunderstood and underutilized, even as the C++ language and development community have advanced. In C++ Templates, Second Edition, three pioneering C++ experts show why, when, and how to use modern templates to build software thats cleaner, faster, more efficient, and easier to maintain.**  **Now extensively updated for the C++11, C++14, and C++17 standards, this new edition presents state-of-the-art techniques for a wider spectrum of applications. The authors provide authoritative explanations of all new language features that either improve templates or interact with them, including variadic templates, generic lambdas, class template argument deduction, compile-time if, forwarding references, and user-defined literals.**  **They also deeply delve into fundamental language concepts (like value categories) and fully cover all standard type traits. The book starts with an insightful tutorial on basic concepts and relevant language features. The remainder of the book serves as a comprehensive reference, focusing first on language details and then on coding techniques, advanced applications, and sophisticated idioms.**  **Throughout, examples clearly illustrate abstract concepts and demonstrate best practices for exploiting all that C++ templates can do. Understand exactly how templates behave, and avoid common pitfalls Use templates to write more efficient, flexible, and maintainable software Master todays most effective idioms and techniques Reuse source code without compromising performance or safety Benefits.** |