**Abstraction (Class/UML)**: store code in structure or function and that structure or function multiple time in our work, avoid duplication. UML is the format of a class, where it consist of name, type and attributes, operators with return type and parameters **Encapsulation**: integration of data and logic within a class implementation **Inheritance**: One class inherits the structure of another class **Polymorphism**: A single interface provides multiple implementation **Preprocessor**: interprets all directives creating a single translation unit for the compiler (insert the contents of all #include header and substitutes all #define) **Compiler**: compiles each translation unit separately and creates a corresponding binary version **Linker**: assembles the various binary units along with the system binaries to create one complete executable binary **Syntactics Errors**: minor error that exist during coding **Semantic Errors**: error that break the rules of coding **Declaration**: create variable **Definition**: create function **References**: an alias for a variable or object **Static Memory**: operating system allocates for the application at load time **Dynamic Memory**: memory that an application obtains from the operating system (allocates and deallocate at run-time) Memory Allocation/Deallocation: pointer = new Type[size] / delete[] pointer / pointer = nullptr **Memory Leak**: application loses the address of dynamically allocated memory (pointer goes out of scoop) **Queries**: report the state of object(const) **Modifier**: change the state of the object **Special**: create, assign and destroy an object **Struct** define a class a public by default, **Class** define a class a private by default **Width**: set place for displaying result (width(10) = \*\*\*\*\*\*\*\*9) **Fill(char)**: set padding charater (usually go with width) **Setf**: to set the position of number, come with left, right, precision(x), scientific, fixed **Constructor**: mechanism for initializing data members at creation-time (if not define value, data members will be all trash) **Destructor**: mechanism for tidying up at end of object’s timeline, deallocate memory before going out of scope **Constructors** created in order of object reference and **Destruct** it in opposite order **Explicit**: access the client code **Implicit**: Access the instance variable **This**: use to distinguish with object that stored data in class/struct, \*this referred to all the object in class  





