CS304
Object Oriented Programming

Important mcqs

Lec 1 - INTRODUCTION

1. **What is the purpose of an introduction?**
   A) To summarize the entire work
   B) To provide a roadmap for the reader
   C) To include all the details of the content
   D) To present the conclusion first
   Answer: B

   Which of the following is a characteristic of a good introduction?
   A) Lengthy and repetitive
   B) Unclear and confusing
   C) Concise and informative
   D) Irrelevant to the topic
   Answer: C

   What does an introduction provide for the reader?
   A) A summary of the conclusion
   B) An overview of the entire work
   C) The main body of the content
   D) None of the above
   Answer: B

   What should be the tone of an introduction?
   A) Informative
   B) Persuasive
   C) Formal
   D) All of the above
   Answer: A

   Which of the following is NOT an objective of an introduction?
   A) To provide context
   B) To grab the reader's attention
   C) To present the conclusion first
   D) To introduce the topic
   Answer: C

   What is the ideal length of an introduction?
   A) Half of the total work
   B) One third of the total work
   C) One quarter of the total work

D) It depends on the length of the work
Answer: C

Which of the following should be included in an introduction?
A) All the details of the content
B) A thesis statement
C) The entire conclusion
D) None of the above
Answer: B

What should be the structure of an introduction?
A) Chronological
B) Random
C) Logical
D) None of the above
Answer: C

What is the main function of an introduction?
A) To provide a conclusion
B) To grab the reader's attention
C) To provide context
D) None of the above
Answer: C

Which of the following is a common mistake to avoid in an introduction?
A) Being too general
B) Being too specific
C) Being too formal
D) All of the above
Answer: A

Lec 2 - INFORMATION HIDING

1. **What is a model?**
   A. A simplified representation of a complex system
   B. A complicated representation of a simple system
   C. A physical replica of a real-world phenomenon
   D. A mathematical equation used in statistics
   Answer: A

   What is the main purpose of a model?
   A. To perfectly capture the complexity of reality
   B. To simulate real-world phenomena
   C. To create a physical replica of a system
   D. To replace the need for experimentation
   Answer: B

   Which fields use models?
   A. Science and engineering
   B. Economics and finance
   C. Computer science and technology
   D. All of the above
   Answer: D

   What are the limitations of a model?
   A. It can make assumptions that may affect its accuracy
   B. It can only capture the complexity of reality perfectly
   C. It is always expensive to develop
   D. It cannot be used to inform decision-making
   Answer: A

   What is an example of a model?
   A. A physical replica of a car
   B. A computer program simulating traffic flow
   C. A mathematical equation representing the weather
   D. All of the above
   Answer: D

   Why are models useful in science?
   A. They allow for the testing of hypotheses
   B. They replace the need for experimentation
   C. They can perfectly capture the complexity of reality
   D. They are always more accurate than real-world data
   Answer: A

   What is the purpose of a mathematical model?
   A. To simulate real-world phenomena
   B. To create a physical replica of a system
   C. To make assumptions about a system
   D. To predict outcomes and inform decision-making
   Answer: D

   What is a disadvantage of using a physical model?
   A. It is always cheaper to develop than other types of models

B. It can be difficult to accurately replicate a real-world phenomenon
C. It cannot be used to test hypotheses
D. It is not suitable for informing decision-making
Answer: B

What is the difference between a model and a theory?
A. A theory is a type of model
B. A model is a type of theory
C. A theory is a well-established explanation for a phenomenon, while a model is a simplified representation of a system
D. A model is more accurate than a theory
Answer: C

How can models be improved?
A. By incorporating more complex variables
B. By reducing the number of assumptions made
C. By including real-world data
D. All of the above
Answer: D

Lec 3 - ABSTRACTION

1. **What is abstraction?**
   A) A process of creating detailed representations of an object
   B) A process of simplifying complex ideas by removing unnecessary details
   C) A process of defining precise specifications of a system
   D) A process of optimizing code for performance
   Answer: B) A process of simplifying complex ideas by removing unnecessary details

   Which field does abstraction play a crucial role in?
   A) Mathematics
   B) Computer science
   C) Art
   D) All of the above
   Answer: D) All of the above

   In computer science, what does abstraction refer to?
   A) A process of hiding implementation details while exposing essential functionality
   B) A process of creating complex algorithms
   C) A process of testing software for bugs
   D) A process of designing user interfaces
   Answer: A) A process of hiding implementation details while exposing essential functionality

   What is the benefit of abstraction in computer science?
   A) More efficient and maintainable software systems
   B) More complex and difficult to understand software systems
   C) Faster software development
   D) Higher quality software systems
   Answer: A) More efficient and maintainable software systems

   Which art form utilizes abstraction to represent the essence of a subject?
   A) Realism
   B) Impressionism
   C) Abstract expressionism
   D) Surrealism
   Answer: C) Abstract expressionism

   What is the cognitive process involved in abstraction?
   A) Creating detailed representations of an object
   B) Simplifying complex ideas by removing unnecessary details
   C) Analyzing complex systems
   D) Memorizing information
   Answer: B) Simplifying complex ideas by removing unnecessary details

   Which of the following is an example of abstraction in mathematics?
   A) Using variables to represent unknown values in equations
   B) Solving complex equations without simplification
   C) Graphing equations without labeling the axes
   D) Using only whole numbers in calculations
   Answer: A) Using variables to represent unknown values in equations

   What is the purpose of abstraction in philosophy?
   A) To understand the nature of reality

B) To create complex arguments
C) To study the history of philosophy
D) To memorize philosophical theories
Answer: A) To understand the nature of reality

Which of the following is NOT a benefit of abstraction?
A) More efficient and maintainable software systems
B) Simplified understanding of complex systems
C) Increased complexity of systems
D) Improved problem-solving ability
Answer: C) Increased complexity of systems

Which term refers to the level of abstraction that focuses on the essential features of a system?
A) High-level abstraction
B) Low-level abstraction
C) Mid-level abstraction
D) No abstraction
Answer: A) High-level abstraction

Lec 4 - CONCEPTS RELATED WITH INHERITANCE

1. **In object-oriented programming, what is inheritance?**
   a) A way to create new objects based on existing ones
   b) A way to create new classes based on existing ones
   c) A way to create new methods based on existing ones
   d) A way to create new properties based on existing ones
   Answer: b) A way to create new classes based on existing ones

   What is a superclass in inheritance?
   a) A class that inherits from another class
   b) A class that is inherited by another class
   c) A class that has no parent class
   d) A class that is created from scratch
   Answer: b) A class that is inherited by another class

   What is a subclass in inheritance?
   a) A class that inherits from another class
   b) A class that is inherited by another class
   c) A class that has no parent class
   d) A class that is created from scratch
   Answer: a) A class that inherits from another class

   What is the purpose of inheritance in object-oriented programming?
   a) To create new objects
   b) To reduce code redundancy
   c) To create new methods
   d) To create new properties
   Answer: b) To reduce code redundancy

   Which of the following keywords is used to denote inheritance in Java?
   a) extends
   b) implements
   c) interface
   d) abstract
   Answer: a) extends

   Which type of inheritance allows a subclass to inherit from multiple parent classes?
   a) Single inheritance
   b) Multiple inheritance
   c) Multilevel inheritance
   d) Hierarchical inheritance
   Answer: b) Multiple inheritance

   Inheritance promotes which principle of object-oriented programming?
   a) Encapsulation
   b) Polymorphism
   c) Abstraction
   d) All of the above
   Answer: d) All of the above

   Which access modifier in Java allows a subclass to access a protected member of its

parent class?
a) public
b) private
c) protected
d) default
Answer: c) protected

What is method overriding in inheritance?
a) Creating a new method in the subclass with the same name as a method in the parent class
b) Creating a new method in the parent class with the same name as a method in the subclass
c) Renaming a method in the subclass to match a method in the parent class
d) Hiding a method in the parent class from the subclass
Answer: a) Creating a new method in the subclass with the same name as a method in the parent class

What is method overloading in inheritance?
a) Creating a new method in the subclass with the same name as a method in the parent class
b) Creating a new method in the parent class with the same name as a method in the subclass
c) Renaming a method in the subclass to match a method in the parent class
d) Creating a new method in the subclass with the same name but different parameters as a method in the parent class
Answer: d) Creating a new method in the subclass with the same name but different parameters as a method in the parent class

Lec 5 - SIMPLE ASSOCIATION

1. **In simple association, how many classes are involved?**
   A) One
   B) Two
   C) Three
   D) Four
   Answer: B) Two

   How is simple association represented in UML diagrams?
   A) As a solid line
   B) As a dotted line
   C) As a dashed line
   D) As a double line
   Answer: A) As a solid line

   In simple association, which class is usually responsible for initiating the interaction?
   A) The class on the left side of the association
   B) The class on the right side of the association
   C) Both classes can initiate the interaction
   D) Neither class can initiate the interaction
   Answer: A) The class on the left side of the association

   Which of the following is an example of simple association?
   A) A car has an engine
   B) A car is a vehicle
   C) A car belongs to a person
   D) A car drives on a road
   Answer: A) A car has an engine

   Which of the following is true about simple association?
   A) It is always one-way
   B) It is always bidirectional
   C) It can be one-way or bidirectional
   D) It is always represented by an arrow
   Answer: C) It can be one-way or bidirectional

   What is the role of the class on the right side of the association in simple association?
   A) To provide functionality to the class on the left side
   B) To receive functionality from the class on the left side
   C) To initiate the interaction with the class on the left side
   D) To define the type of the association
   Answer: B) To receive functionality from the class on the left side

   Which of the following is not an example of simple association?
   A) A dog has a tail
   B) A book belongs to a library
   C) A bird can fly
   D) A student attends a class
   Answer: C) A bird can fly

   Which of the following is not true about simple association?
   A) It is a type of relationship between classes

B) It enables communication and collaboration between classes
C) It is always one-to-one
D) It can be one-way or bidirectional
Answer: C) It is always one-to-one

What is the difference between simple association and inheritance?
A) Inheritance involves a parent-child relationship, while simple association does not
B) Simple association involves a parent-child relationship, while inheritance does not
C) Simple association enables communication and collaboration between classes, while inheritance does not
D) Inheritance and simple association are the same thing
Answer: A) Inheritance involves a parent-child relationship, while simple association does not

Which of the following is an example of bidirectional simple association?
A) A person has a car
B) A car belongs to a person
C) A teacher teaches a class
D) A student attends a class
Answer: A) A person has a car and a car belongs to a person can both be examples of bidirectional simple association.

Lec 6 - CLASS COMPATIBILITY

1. **What is class compatibility in object-oriented programming?**
   a) The ability of classes to be used interchangeably
   b) The ability of one class to use objects of another class without errors
   c) The ability of classes to be inherited from each other
   d) The ability of classes to have the same name
   Answer: b) The ability of one class to use objects of another class without errors

   Which of the following can affect class compatibility?
   a) Inheritance
   b) Interfaces
   c) Method signatures
   d) All of the above
   Answer: d) All of the above

   What is the difference between static and dynamic class compatibility?
   a) Static compatibility is checked at compile time, while dynamic compatibility is checked at runtime
   b) Static compatibility is checked at runtime, while dynamic compatibility is checked at compile time
   c) There is no difference between static and dynamic class compatibility
   d) Static compatibility is a type of inheritance, while dynamic compatibility is a type of polymorphism
   Answer: a) Static compatibility is checked at compile time, while dynamic compatibility is checked at runtime

   Which of the following is an example of static class compatibility?
   a) An object of a subclass being passed to a method that expects an object of the superclass
   b) An object of a class implementing an interface being passed to a method that expects an object of the interface
   c) An object of a class with a compatible method signature being passed to a method
   d) None of the above
   Answer: c) An object of a class with a compatible method signature being passed to a method

   Which of the following is an example of dynamic class compatibility?
   a) A superclass reference being used to refer to an object of a subclass
   b) An interface reference being used to refer to an object of a class implementing the interface
   c) A method being overridden in a subclass
   d) All of the above
   Answer: a) A superclass reference being used to refer to an object of a subclass

   What is type checking in relation to class compatibility?
   a) The process of checking if a variable or object is of a specific type
   b) The process of checking if two classes are compatible
   c) The process of checking if an object can be cast to a specific type
   d) The process of checking if a method signature is compatible with a class
   Answer: a) The process of checking if a variable or object is of a specific type

   What is casting in relation to class compatibility?
   a) The process of checking if a variable or object is of a specific type

b) The process of checking if two classes are compatible
c) The process of converting an object to a different type
d) The process of checking if a method signature is compatible with a class
Answer: c) The process of converting an object to a different type

What happens if an object is cast to an incompatible type?
a) An exception is thrown at runtime
b) The object is automatically converted to the compatible type
c) The object remains unchanged
d) None of the above
Answer: a) An exception is thrown at runtime

Which of the following is an example of a type casting error?
a) Casting an object of a subclass to a superclass type
b) Casting an object of an interface implementation to an interface type
c) Casting an object to a different type than it was created as
d) All of the above
Answer: d) All of the above

Why is class compatibility important in software development?
a) It ensures that classes can work together effectively
b) It makes software systems easier to maintain
c) It reduces the likelihood of errors and bugs
d) All

Lec 7 - CLASS

1. **What does CLASS stand for?**
   a) Classroom Assessment and Scoring System
   b) Children's Learning and Support System
   c) Classroom Analysis and Skills System
   d) Creative Learning and Assessment Strategies
   Answer: a) Classroom Assessment and Scoring System

   How many domains does CLASS evaluate?
   a) 2
   b) 3
   c) 4
   d) 5
   Answer: c) 4

   Which of the following is not a domain evaluated by CLASS?
   a) Emotional Support
   b) Classroom Organization
   c) Instructional Support
   d) Student Performance
   Answer: d) Student Performance

   What is the purpose of CLASS?
   a) To measure and improve the quality of teacher-student interactions in early childhood education settings
   b) To assess student academic achievement
   c) To evaluate teacher performance
   d) To promote student engagement in the classroom
   Answer: a) To measure and improve the quality of teacher-student interactions in early childhood education settings

   Who developed the CLASS tool?
   a) The National Institute for Early Education Research
   b) The National Association for the Education of Young Children
   c) The University of Virginia
   d) The American Educational Research Association
   Answer: c) The University of Virginia

   How many dimensions are evaluated within the Emotional Support domain of CLASS?
   a) 2
   b) 3
   c) 4
   d) 5
   Answer: b) 3

   Which dimension within the Classroom Organization domain of CLASS evaluates how well teachers manage student behavior?
   a) Behavior Management
   b) Productivity

c) Instructional Learning Formats
d) Concept Development
Answer: a) Behavior Management

Which dimension within the Instructional Support domain of CLASS evaluates how well teachers promote higher-order thinking skills?
a) Quality of Feedback
b) Cognitive Complexity
c) Language Modeling
d) Concept Development
Answer: b) Cognitive Complexity

How is CLASS typically used in early childhood education settings?
a) To evaluate student academic achievement
b) To assess teacher performance
c) To identify areas for improvement in teacher-student interactions
d) To promote student engagement in the classroom
Answer: c) To identify areas for improvement in teacher-student interactions

Which of the following is not a benefit of using CLASS in early childhood education settings?
a) Provides feedback to teachers to improve their practice
b) Supports professional development
c) Enhances student academic achievement
d) Promotes a positive classroom climate
Answer: c) Enhances student academic achievement

Lec 8 - MEMBER FUNCTIONS

1. **What is the primary purpose of member functions in object-oriented programming?**
   A. To declare variables
   B. To initialize objects
   C. To perform operations on objects
   D. To create new objects
   Answer: C

   Which keyword is used to define a member function in C++?
   A. class
   B. function
   C. method
   D. this
   Answer: C

   Which type of member function can access private data members of a class?
   A. Public
   B. Private
   C. Protected
   D. Friend
   Answer: B

   Which type of member function does not have access to the this pointer?
   A. Static
   B. Virtual
   C. Inline
   D. Friend
   Answer: A

   Which keyword is used to call a member function on an object in C++?
   A. new
   B. delete
   C. this
   D. dot operator (.)
   Answer: D

   Which type of member function is used to initialize an object with a set of default values?
   A. Constructor
   B. Destructor
   C. Virtual function
   D. Operator function
   Answer: A

   Which type of member function is called automatically when an object is destroyed?
   A. Constructor
   B. Destructor
   C. Virtual function
   D. Operator function
   Answer: B

   Which type of member function can be called without creating an object of the class?
   A. Constructor

B. Destructor
C. Static function
D. Friend function
Answer: C

Which keyword is used to access a data member of a class within a member function?
A. private
B. public
C. protected
D. this
Answer: D

Which type of member function is used to overload operators such as +, -, *, and /?
A. Constructor
B. Destructor
C. Virtual function
D. Operator function
Answer: D

Lec 9 - SHALLOW COPY

1. **What is shallow copy?**
   a) A copy of the data itself
   b) A copy of the pointers or references to the data members
   c) A copy of the entire object
   Answer: b

   What happens when changes are made to the data in a shallow copy?
   a) The original object is also changed
   b) The original object remains unchanged
   c) The new object is destroyed
   Answer: a

   Which type of copy is a shallow copy?
   a) A deep copy
   b) A partial copy
   c) A pointer copy
   Answer: c

   What is the purpose of shallow copying?
   a) To create a new object with the same data as the original object
   b) To create a copy of the original object
   c) To create a reference to the original object
   Answer: c

   Can a shallow copy be modified without affecting the original object?
   a) Yes
   b) No
   Answer: b

   Which programming languages support shallow copying by default?
   a) Java
   b) Python
   c) C++
   Answer: c

   What is the difference between a shallow copy and a deep copy?
   a) A shallow copy only copies pointers, while a deep copy copies the entire object
   b) A shallow copy copies the entire object, while a deep copy only copies pointers
   c) There is no difference between the two
   Answer: a

   Is it possible to create a shallow copy manually in C++?
   a) Yes
   b) No
   Answer: a

   What happens if a shallow copy is deleted before the original object?
   a) The original object is deleted

b) The new object is deleted
c) Both the original and new objects are deleted
Answer: b

Can a shallow copy be used to create an independent copy of an object?
a) Yes
b) No
Answer: b

Lec 10 - USES OF THIS POINTER

1. **In object-oriented programming, what is the "this" pointer?**
   a) A reference to the object that is currently being operated on
   b) A reference to the parent object
   c) A reference to the child object
   d) A reference to the base class
   Answer: a) A reference to the object that is currently being operated on

   What is the main use of the "this" pointer?
   a) To access member variables or functions of the current object
   b) To access member variables or functions of another object
   c) To create a new object
   d) To destroy an object
   Answer: a) To access member variables or functions of the current object

   Can the "this" pointer be used to access member variables of other objects of the same class?
   a) Yes
   b) No
   Answer: b) No

   Can the "this" pointer be used to pass the object as an argument to another function?
   a) Yes
   b) No
   Answer: a) Yes

   Can the "this" pointer be used to return the object from a function?
   a) Yes
   b) No
   Answer: a) Yes

   What is the benefit of using the "this" pointer?
   a) It helps to differentiate between multiple objects of the same class
   b) It helps to create new objects
   c) It helps to destroy objects
   d) It helps to access variables of other classes
   Answer: a) It helps to differentiate between multiple objects of the same class

   Is the "this" pointer supported by all programming languages?
   a) Yes
   b) No
   Answer: b) No

   In C++, what is the syntax for using the "this" pointer to access a member variable?
   a) this.memberVariable
   b) memberVariable.this
   c) this->memberVariable
   d) memberVariable->this
   Answer: c) this->memberVariable

   Can the "this" pointer be used outside of a member function?
   a) Yes

b) No
Answer: b) No

Is the "this" pointer a constant or a variable?
a) Constant
b) Variable
Answer: a) Constant

Lec 11 - USAGE EXAMPLE OF CONSTANT MEMBER FUNCTIONS

1. **In which scenario would you use a constant member function?**
   a. When you want to modify the object
   b. When you want to ensure that the object cannot be modified
   c. When you want to improve performance
   d. Both b and c
   Answer: d

   Which of the following is an example of a scenario where constant member functions would be useful?
   a. Implementing a class that represents a car
   b. Implementing a class that represents a mathematical vector
   c. Implementing a class that represents a text editor
   d. Implementing a class that represents a video game character
   Answer: b

   Can a constant member function modify the state of the object it is called on?
   a. Yes
   b. No
   Answer: b

   What is the benefit of using constant member functions?
   a. They allow you to modify the object
   b. They improve performance
   c. They ensure that the object cannot be modified
   d. They allow you to access private member variables
   Answer: c

   Which keyword is used to declare a member function as constant?
   a. const
   b. static
   c. virtual
   d. volatile
   Answer: a

   Which of the following is an example of a constant member function for a class representing a mathematical vector?
   a. void setX(double x)
   b. double getX() const
   c. double length()
   d. void normalize()
   Answer: b

   What is the purpose of a constant member function for a class representing a mathematical vector?
   a. To modify the vector
   b. To return the length of the vector
   c. To normalize the vector
   d. To ensure that the vector cannot be modified
   Answer: d

   Which of the following is an example of a scenario where constant member functions

would not be useful?
a. Implementing a class that represents a bank account
b. Implementing a class that represents a calendar event
c. Implementing a class that represents a temperature sensor
d. Implementing a class that represents a musical instrument
Answer: d

What is the return type of a constant member function?
a. void
b. int
c. double
d. Depends on the implementation
Answer: d

Can you call a non-constant member function from a constant member function?
a. Yes
b. No
Answer: b

Lec 12 - ACCESSING STATIC DATA MEMBER

1. **What is a static data member in C++?**
   a) A data member that can only be accessed by member functions
   b) A data member that can be accessed by any function or method within the class
   c) A data member that is unique to each instance of a class
   d) A data member that is declared using the const keyword
   Answer: b

   How is a static data member declared in C++?
   a) Using the const keyword
   b) Using the static keyword
   c) Using the public keyword
   d) Using the friend keyword
   Answer: b

   How is a static data member accessed in C++?
   a) Using the object name followed by the dot operator
   b) Using the object name followed by the arrow operator
   c) Using the class name followed by the dot operator
   d) Using the class name followed by the arrow operator
   Answer: c

   Which of the following statements is true about static data members?
   a) They are unique to each instance of a class
   b) They can only be accessed by member functions
   c) They are shared among all objects of a class
   d) They are declared using the const keyword
   Answer: c

   What is the default value of a static data member in C++?
   a) 0
   b) 1
   c) Null
   d) Undefined
   Answer: a

   What is the scope of a static data member in C++?
   a) Global scope
   b) Local scope
   c) Class scope
   d) Namespace scope
   Answer: c

   What is the lifetime of a static data member in C++?
   a) Until the end of the program
   b) Until the end of the function in which it is declared
   c) Until the object is destroyed
   d) Until it is explicitly deleted
   Answer: a

   How many instances of a static data member are there in a class?
   a) One for each instance of the class

b) One for all instances of the class
c) One for each member function
d) None of the above
Answer: b

Which keyword is used to access a static data member outside the class in C++?
a) private
b) public
c) static
d) friend
Answer: c

Can a static data member be modified by a non-static member function?
a) Yes
b) No
Answer: a

Lec 13 - POINTER TO OBJECTS

1. **What is a pointer to an object in C++?**
   a) A variable that stores the value of an object
   b) A variable that stores the memory address of an object
   c) A variable that stores the size of an object
   d) A variable that stores the name of an object
   Answer: b) A variable that stores the memory address of an object.

   What is the syntax to declare a pointer to an object in C++?
   a) int ptr;
   b) obj pointer;
   c) obj *ptr;
   d) obj -> pointer;
   Answer: c) obj *ptr;

   How is the value of an object pointed to by a pointer accessed in C++?
   a) Using the * operator
   b) Using the & operator
   c) Using the -> operator
   d) Using the . operator
   Answer: a) Using the * operator.

   What is the purpose of using pointers to objects in C++?
   a) To dynamically allocate memory for objects
   b) To pass objects to functions by reference
   c) To manipulate objects indirectly
   d) All of the above
   Answer: d) All of the above.

   Can a pointer to an object be null in C++?
   a) Yes
   b) No
   Answer: a) Yes.

   What is the difference between a pointer to an object and a reference to an object in C++?
   a) A pointer can be null, while a reference cannot.
   b) A pointer can be reassigned to point to a different object, while a reference cannot.
   c) A pointer requires the * operator to access the object's value, while a reference does not.
   d) All of the above.
   Answer: d) All of the above.

   How is memory allocated for an object pointed to by a pointer in C++?
   a) Using the new operator
   b) Using the delete operator
   c) Using the malloc function
   d) Using the free function
   Answer: a) Using the new operator.

   What is the purpose of the -> operator in C++?
   a) To access a member of a class or structure pointed to by a pointer

b) To declare a pointer to an object
c) To declare a reference to an object
d) None of the above
Answer: a) To access a member of a class or structure pointed to by a pointer.

How can a pointer to an object be passed to a function in C++?
a) By value
b) By reference
c) By const reference
d) All of the above
Answer: b) By reference.

What is a dangling pointer in C++?
a) A pointer that points to a null object
b) A pointer that points to an object that has been deleted or deallocated
c) A pointer that points to a new object
d) A pointer that points to an object that has not been initialized
Answer: b) A pointer that points to an object that has been deleted or deallocated.

Lec 14 - COMPOSITION

1. **What is composition in object-oriented programming?**
   a) A way of creating complex objects by combining simpler objects or data types
   b) A way of inheriting properties and behaviors from a parent class
   c) A way of creating objects from a template or blueprint
   Answer: a

   In composition, which class contains an instance of another class as a member variable?
   a) The parent class
   b) The child class
   c) Both classes
   Answer: b

   Which symbol is used to denote composition in a UML class diagram?
   a) A solid line with an arrow pointing to the contained class
   b) A dashed line with an arrow pointing to the contained class
   c) A solid line connecting the two classes
   Answer: b

   What is the purpose of using composition in object-oriented programming?
   a) To create objects with complex behavior
   b) To simplify the implementation of inheritance
   c) To create objects with a strong is-a relationship
   Answer: a

   How does composition differ from inheritance?
   a) Composition is a type of association between classes, while inheritance is a way of inheriting properties and behaviors from a parent class
   b) Composition is used to create a strong is-a relationship between classes, while inheritance is used to combine behaviors from multiple classes
   c) Composition and inheritance are identical concepts
   Answer: a

   Which keyword is used to define a composition relationship in C++?
   a) extends
   b) implements
   c) none
   Answer: c

   What happens to the contained object when the containing object is destroyed in composition?
   a) The contained object is destroyed automatically
   b) The contained object remains alive
   c) It depends on how the composition is implemented
   Answer: a

   Can a class have multiple instances of another class as member variables in composition?
   a) Yes

b) No
c) It depends on the programming language being used
Answer: a

Which of the following is an example of composition in real-world objects?
a) A car's engine and transmission
b) A car and a truck
c) A car's tires and brakes
Answer: a

Which of the following is not a benefit of using composition in object-oriented programming?
a) Encapsulation of behavior and data
b) Code reuse
c) Simplified implementation of inheritance
Answer: c

Lec 15 - AGGREGATION

1. **What is aggregation in object-oriented programming?**
   a. A type of inheritance
   b. A way of creating complex objects by combining simpler objects
   c. A type of association between classes where one class contains a collection of another class's objects
   Answer: c

   Can the contained objects in aggregation exist independently of the containing object?
   a. Yes
   b. No
   Answer: a

   How is aggregation represented in a UML class diagram?
   a. With a solid line and an arrow pointing to the contained class
   b. With a dashed line and an arrow pointing to the contained class
   c. With a dotted line and an arrow pointing to the contained class
   Answer: b

   What is the purpose of using aggregation in object-oriented programming?
   a. To create complex objects by combining simpler objects
   b. To inherit properties and behaviors from a parent class
   c. To encapsulate behavior and data
   Answer: a

   Can a class have multiple instances of another class as member variables in aggregation?
   a. Yes
   b. No
   Answer: a

   How does aggregation differ from composition?
   a. In aggregation, the contained objects cannot exist independently of the containing object
   b. In composition, the contained objects can exist independently of the containing object
   c. There is no difference between aggregation and composition
   Answer: b

   Can the contained objects be shared among multiple containing objects in aggregation?
   a. Yes
   b. No
   Answer: a

   What happens to the contained objects when the containing object is destroyed in aggregation?
   a. The contained objects are automatically destroyed
   b. The contained objects continue to exist independently of the containing object
   c. It depends on the implementation
   Answer: b

   How does aggregation support code reuse?
   a. By allowing for the creation of complex objects by combining simpler objects

b. By inheriting properties and behaviors from a parent class
c. By encapsulating behavior and data
Answer: a

What are some real-world examples of aggregation?
a. A car's engine and transmission
b. A house's rooms and furniture
c. A human's body parts
Answer: b

Lec 16 - OPERATOR OVERLOADING

1. **Which keyword is used to overload operators in C++?**
   a. override
   b. overload
   c. friend
   d. operator
   Answer: d

   Which of the following operators cannot be overloaded in C++?
   a. +
   b. &&
   c. =
   d. ->
   Answer: d

   Which of the following is a unary operator?
   a. +
   b. ++
   c. <<
   d. -
   Answer: d

   Which of the following is used to overload the subscript operator?
   a. []
   b. ()
   c. {}
   d. <>
   Answer: a

   Which of the following is used to overload the insertion operator for cout?
   a. <<
   b. >>
   c. ::
   d. ->
   Answer: a

   Which of the following is used to overload the prefix increment operator?
   a. ++
   b. --
   c. ==
   d. /
   Answer: a

   Which of the following operators has the highest **precedence**?
   a. ||
   b. *
   c. !=
   d. +
   Answer: b

   Which of the following is a binary operator?
   a. !

b. ^
c. ~
d. &&
Answer: b

Which of the following is a comparison operator?
a. !
b. +
c. <
d. &
Answer: c

Which of the following is used to overload the addition operator?
a. &&
b. |
c. ^
d. +
Answer: d

Lec 17 - OVERLOADING ASSIGNMENT OPERATOR

1. **What is the syntax for overloading the assignment operator in C++?**
   A) operator = ()
   B) operator ()
   C) operator ()
   D) operator += ()
   Answer: A

   Which of the following is a valid signature for an overloaded assignment operator that takes a reference to the class object as a parameter?
   A) MyClass operator=(MyClass& obj)
   B) void operator=(const MyClass& obj)
   C) MyClass& operator=(const MyClass& obj)
   D) void operator=(MyClass& obj)
   Answer: C

   What is the return type of an overloaded assignment operator?
   A) void
   B) the class type
   C) int
   D) bool
   Answer: B

   How is the overloaded assignment operator invoked in C++?
   A) using the = operator
   B) using the copy constructor
   C) using the constructor
   D) using the destructor
   Answer: A

   What is the purpose of overloading the assignment operator in C++?
   A) to allow objects of a class to be assigned values using the = operator
   B) to allow objects of a class to be compared using the == operator
   C) to allow objects of a class to be initialized using the constructor
   D) to allow objects of a class to be destroyed using the destructor
   Answer: A

   Which of the following is true about the copy constructor and the assignment operator?
   A) they both take a reference to the class object as a parameter
   B) they both return a reference to the class object
   C) they both perform a shallow copy of the object's data members
   D) they both perform a deep copy of the object's data members
   Answer: D

   What is the difference between the copy constructor and the assignment operator?
   A) the copy constructor creates a new object, while the assignment operator modifies an existing object
   B) the copy constructor takes a const reference to the object, while the assignment operator takes a non-const reference

C) the copy constructor performs a deep copy of the object, while the assignment operator performs a shallow copy
D) the copy constructor returns a reference to the object, while the assignment operator returns void
Answer: A

How do you avoid issues with self-assignment when overloading the assignment operator?
A) by checking for self-assignment using the == operator
B) by using a copy constructor to create a new object and then assigning it to the existing object
C) by checking for self-assignment using the this pointer
D) by using a swap function to swap the contents of the object with a temporary object
Answer: D

Which of the following is a common practice when overloading the assignment operator?
A) returning a copy of the object from the function
B) returning a reference to the object from the function
C) using dynamic memory allocation to perform a deep copy of the object's data members
D) using the default implementation of the operator provided by the compiler
Answer: C

Which of the following is true about the return type of the overloaded assignment operator?
A) it must be a built-in type such as int or bool
B) it can be any user-defined type
C) it must be the same type as the class being overloaded
D) it can be a different type from the class being overloaded
Answer: C

Lec 18 - SELF ASSIGNMENT PROBLEM

1. **What is self-assignment?**
   a) Assigning a pointer to a different object
   b) Assigning an object to itself
   c) Assigning a value to a constant variable
   d) Assigning a value to a variable of a different data type
   Answer: b

   What can happen if self-assignment is not handled properly?
   a) Memory leaks
   b) Undefined behavior
   c) Corruption of the object's data
   d) All of the above
   Answer: d

   Which operator is commonly affected by the self-assignment problem?
   a) Comparison operator
   b) Unary operator
   c) Binary operator
   d) Assignment operator
   Answer: d

   What is a common technique for handling self-assignment in the assignment operator?
   a) Copying the object to a temporary object before performing the copy
   b) Checking if the object being assigned is the same as the original object before performing the copy
   c) Swapping the object with a copy of itself
   d) None of the above
   Answer: b

   What is the purpose of handling self-assignment in the assignment operator?
   a) To avoid memory leaks
   b) To prevent undefined behavior
   c) To ensure proper functioning of the program
   d) All of the above
   Answer: d

   Which of the following is a potential issue that can arise from self-assignment?
   a) Data corruption
   b) Memory leaks
   c) Undefined behavior
   d) All of the above
   Answer: d

   Why is it important to properly handle self-assignment in the assignment operator?
   a) To prevent crashes
   b) To avoid undefined behavior
   c) To optimize program performance
   d) All of the above
   Answer: b

   How can self-assignment be checked in the assignment operator?

b) Using a conditional statement
c) Using a loop
d) None of the above
Answer: b

Which of the following is an example of self-assignment?
a) a = b
b) a = a
c) a = &b
d) a = *b
Answer: b

What can happen if self-assignment is not handled properly in a program?
a) The program may crash
b) The program may behave unpredictably
c) The program may run slower than expected
d) All of the above
Answer: d

Lec 19 - STREAM INSERTION OPERATOR

1. **What is the return type of the stream insertion operator (<<)?**
   a) void
   b) istream&
   c) ostream&
   d) string
   Answer: c) ostream&

   Which of the following is an example of overloading the stream insertion operator for a custom class?
   a) int x; cin >> x;
   b) cout << "Hello, World!" << endl;
   c) cout << obj;
   d) cin << obj;
   Answer: c) cout << obj;

   Which of the following is an example of using the stream insertion operator to output multiple values?
   a) cout << "The value of x is " << x;
   b) cout << "The sum of " << x << " and " << y << " is " << x + y;
   c) cout << "Enter a value: ";
   d) cout << "The result is " << result << endl;
   Answer: b) cout << "The sum of " << x << " and " << y << " is " << x + y;

   Which of the following is a possible implementation of the stream insertion operator for a custom class?
   a) ostream& operator<<(ostream& out, MyClass obj) { /* implementation / }
   b) istream& operator<<(istream& in, MyClass obj) { / implementation / }
   c) MyClass& operator<<(MyClass obj) { / implementation / }
   d) void operator<<(MyClass obj) { / implementation / }
   Answer: a) ostream& operator<<(ostream& out, MyClass obj) { / implementation */ }

   Which of the following is a correct syntax for using the stream insertion operator to output an object?
   a) cout << MyClass;
   b) cout << object.MyClass;
   c) object << cout;
   d) cout << object;
   Answer: d) cout << object;

   Which of the following is a correct syntax for overloading the stream insertion operator for a custom class?
   a) void operator<<();
   b) void operator<<(ostream& out);
   c) void operator<<(ostream& out, MyClass obj);
   d) void operator<<(MyClass obj);
   Answer: c) void operator<<(ostream& out, MyClass obj);

   Which of the following is a correct implementation of the stream insertion operator for a

custom class that has private data members?
a) ostream& operator<<(ostream& out, MyClass obj) { out << obj.privateMember; }
b) ostream& operator<<(ostream& out, MyClass obj) { obj.privateMember << out; }
c) ostream& operator<<(ostream& out, MyClass obj) { obj.getPrivateMember() << out; }
d) None of the above.
Answer: c) ostream& operator<<(ostream& out, MyClass obj) { obj.getPrivateMember() << out; }

Which of the following is an example of using the stream insertion operator to output a literal value?
a) cout << "Hello, World!";
b) cout << x;
c) cin >> x;
d) cout << "Enter a value: ";
Answer: a) cout << "Hello, World!";

Which of the following is a correct implementation of the stream insertion operator for a custom class that has a non-static data member?
a) ostream& operator<<(ostream& out, MyClass obj) { obj.dataMember << out; }
b) ostream& operator<<(ostream& out, MyClass obj) { obj.dataMember >> out; }
c) ostream& operator<<(ostream& out, MyClass obj) { out << obj.dataMember; }
d) None of the above.
Answer: c) ostream& operator<<(ostream& out, MyClass obj) { out << obj.dataMember; }

Which of the following is an example of using the stream insertion operator

Lec 20 - SUBSCRIPT [] OPERATOR

1. **Which operator is used to access elements of an array or container class?**
   a) () operator
   b) {} operator
   c) [] operator
   d) -> operator
   Answer: c) [] operator

   What is the parameter type for an overloaded subscript operator function?
   a) int
   b) char
   c) string
   d) Depends on the type of the elements being accessed
   Answer: d) Depends on the type of the elements being accessed

   Which of the following is a valid use of the subscript operator?
   a) accessing the nth character of a string
   b) accessing the nth element of an array
   c) accessing the nth element of a vector
   d) all of the above
   Answer: d) all of the above

   What is the return type of the subscript operator function?
   a) void
   b) int
   c) char
   d) Depends on the type of the elements being accessed
   Answer: d) Depends on the type of the elements being accessed

   Which of the following is true regarding the subscript operator overloading?
   a) Only one overload of the subscript operator is allowed per class.
   b) The overload function must be a member function of the class.
   c) The overload function must be a friend function of the class.
   d) The overload function must take two parameters.
   Answer: b) The overload function must be a member function of the class.

   What is the purpose of subscript operator overloading?
   a) To provide a custom element access behavior for user-defined classes.
   b) To access private data members of a class.
   c) To perform arithmetic operations on array elements.
   d) None of the above.
   Answer: a) To provide a custom element access behavior for user-defined classes.

   Which of the following is a disadvantage of using the subscript operator?
   a) It can lead to out-of-bounds access.
   b) It is slower than pointer arithmetic.
   c) It cannot be used with containers like maps and sets.
   d) It cannot be overloaded for user-defined classes.
   Answer: a) It can lead to out-of-bounds access.

   Which of the following is true regarding the subscript operator overloading for a

container class?

a) The operator function must return a reference to the element being accessed.
b) The operator function must return a copy of the element being accessed.
c) The operator function must take a single parameter of type int.
d) The operator function is not allowed to modify the container.
Answer: a) The operator function must return a reference to the element being accessed.

Which of the following is a valid example of subscript operator overloading?

a) int operator[](int i);
b) void operator[](int i);
c) int& operator[](int i);
d) int* operator[](int i);
Answer: c) int& operator[](int i);

What happens if the subscript operator function returns a copy of the element being accessed?

a) The copy is returned by value and can be modified independently of the original element.
b) The copy is returned by reference and any modifications made to it will affect the original element.
c) The program will not compile.
d) None of the above.
Answer: a) The copy is returned by value and can be modified independently of the original element.

Lec 21 - BEHAVIOR OF ++ AND -- FOR PRE-DEFINED TYPES

What is the result of the following code snippet?

css

Copy code

int a = 5;

int b = a++;

a) a = 5, b = 6

b) a = 6, b = 5

c) a = 5, b = 5

d) a = 6, b = 6

Answer: a) a = 5, b = 6

What is the result of the following code snippet?

css

Copy code

int a = 5;

int b = ++a;

a) a = 5, b = 6

b) a = 6, b = 5

c) a = 5, b = 5

d) a = 6, b = 6

Answer: b) a = 6, b = 6

What is the result of the following code snippet?

css

Copy code

float a = 5.0;

float b = a++;

a) a = 5.0, b = 6.0

b) a = 6.0, b = 5.0

c) a = 5.0, b = 5.0

d) a = 6.0, b = 6.0

Answer: a) a = 6.0, b = 5.0

What is the result of the following code snippet?

css

Copy code

float a = 5.0;

float b = ++a;

a) a = 5.0, b = 6.0

b) a = 6.0, b = 5.0

c) a = 5.0, b = 5.0

d) a = 6.0, b = 6.0

Answer: b) a = 6.0, b = 6.0

What is the result of the following code snippet?

css

Copy code

char a = 'a';

char b = a++;

a) a = 'a', b = 'b'

b) a = 'b', b = 'a'

c) a = 'a', b = 'a'

d) a = 'b', b = 'b'

Answer: a) a = 'b', b = 'a'

What is the result of the following code snippet?

css

Copy code

char a = 'a';

char b = ++a;

a) a = 'a', b = 'b'

b) a = 'b', b = 'a'

c) a = 'a', b = 'a'

d) a = 'b', b = 'b'

Answer: b) a = 'b', b = 'b'

What is the result of the following code snippet?

css

Copy code

int a = 5;

int b = a-- + 3;

a) a = 5, b = 8

b) a = 4, b = 7

c) a = 5, b = 7

d) a = 4, b = 8

Answer: c) a = 4, b = 8

What is the result of the following code snippet?

css

Copy code

```
int a = 5;

int b = --a + 3;
```

a) a = 5, b = 7

b) a = 4, b = 7

c) a = 5, b = 6

d) a = 4, b = 6

Answer: b) a = 4, b = 6

Lec 22 - PRACTICAL IMPLEMENTATION OF INHERITANCE IN C

1. **Inheritance in C can be implemented using:**
   a) Structures and function pointers
   b) Classes and objects
   c) Inheritance keyword
   d) None of the above
   Answer: a) Structures and function pointers

   Which of the following is not a type of inheritance?
   a) Single inheritance
   b) Multiple inheritance
   c) Hierarchical inheritance
   d) Parallel inheritance
   Answer: d) Parallel inheritance

   The derived class inherits:
   a) All the properties and methods of the base class
   b) Only the properties of the base class
   c) Only the methods of the base class
   d) None of the above
   Answer: a) All the properties and methods of the base class

   What is the syntax to define a derived structure in C?
   a) struct Derived : Base {}
   b) struct Derived extends Base {}
   c) struct Derived : public Base {}
   d) struct Derived : private Base {}
   Answer: c) struct Derived : public Base {}

   Inheritance is used to:
   a) Achieve code reusability
   b) Encapsulate data
   c) Control access to data
   d) None of the above
   Answer: a) Achieve code reusability

   Which type of inheritance allows a derived class to inherit from multiple base classes?
   a) Single inheritance
   b) Multiple inheritance
   c) Hierarchical inheritance
   d) Hybrid inheritance
   Answer: b) Multiple inheritance

   Which keyword is used to call the constructor of the base class from the derived class constructor?
   a) super
   b) base
   c) this
   d) parent
   Answer: b) base

   Which type of inheritance involves creating a new class that inherits from a base class,

and then creating another class that inherits from the new class?
a) Single inheritance
b) Multiple inheritance
c) Hierarchical inheritance
d) Hybrid inheritance
Answer: c) Hierarchical inheritance

Which of the following is not a benefit of inheritance?
a) Code reusability
b) Improved maintainability
c) Reduced coupling
d) Increased code complexity
Answer: d) Increased code complexity

Which of the following is an example of polymorphism?
a) Overriding a method in a derived class
b) Calling a method from the base class
c) Inheriting properties from a base class
d) None of the above
Answer: a) Overriding a method in a derived class

Lec 23 - ACCESSING BASE CLASS MEMBER FUNCTIONS IN DERIVED CLASS

1. **In object-oriented programming, a derived class can access base class member functions using:**
   a) The dot operator (.)
   b) The arrow operator (->)
   c) The scope resolution operator (::)
   d) None of the above
   Answer: c) The scope resolution operator (::)

   If a base class member function is declared as private, it can be accessed directly by the derived class.
   a) True
   b) False
   Answer: b) False

   In C++, if a base class member function is declared as protected, it can be accessed by:
   a) The derived class and any other class
   b) The derived class only
   c) Any class within the same namespace
   d) None of the above
   Answer: b) The derived class only

   When accessing a base class member function from a derived class, the derived class can modify the base class member function.
   a) True
   b) False
   Answer: b) False

   When a derived class defines a member function with the same name as a member function in the base class, it is called:
   a) Method overloading
   b) Method overriding
   c) Method shadowing
   d) None of the above
   Answer: b) Method overriding

   In C++, if a base class member function is virtual, it can be overridden by a member function in the derived class.
   a) True
   b) False
   Answer: a) True

   If a derived class inherits from multiple base classes, and both base classes have member functions with the same name, the derived class can access both functions using the scope resolution operator.
   a) True
   b) False
   Answer: a) True

   A derived class can access the private member variables of the base class using the

scope resolution operator.
a) True
b) False
Answer: b) False

If a base class has a constructor with arguments, the derived class must call the base class constructor explicitly in its own constructor.
a) True
b) False
Answer: a) True

In C++, the order in which base classes are specified in a derived class declaration affects the order in which their constructors are called.
a) True
b) False
Answer: a) True

Lec 24 - MODIFIED DEFAULT CONSTRUCTOR

1. **What is a modified default constructor?**
   a) A constructor that takes no arguments
   b) A constructor that is defined by the programmer
   c) A default constructor that has been customized to initialize the class data members to specific values
   d) None of the above
   Answer: c

   When is a default constructor provided by the compiler?
   a) When no other constructors are defined for the class
   b) When a constructor is defined by the programmer
   c) When a class has no data members
   d) None of the above
   Answer: a

   How can a modified default constructor be useful?
   a) It allows the programmer to provide a default state for the class when it is created
   b) It allows the user to provide initialization values for the class data members
   c) It allows the programmer to create multiple instances of the class with different initial values
   d) None of the above
   Answer: a

   Can a modified default constructor take arguments?
   a) Yes
   b) No
   Answer: b

   What happens if a modified default constructor is not defined for a class?
   a) The compiler will provide a default constructor that initializes the class data members to default values
   b) The program will not compile
   c) The class data members will be left uninitialized
   d) None of the above
   Answer: a

   What is the difference between a default constructor and a modified default constructor?
   a) A default constructor initializes the class data members to default values, while a modified default constructor initializes them to specific values
   b) There is no difference, they are the same thing
   c) A default constructor is provided by the compiler, while a modified default constructor is defined by the programmer
   d) None of the above
   Answer: a

   Can a modified default constructor be overloaded?
   a) Yes
   b) No
   Answer: a

   What is the syntax for defining a modified default constructor?
   a) ClassName();

b) ClassName::ClassName();
c) ClassName::ModifiedDefaultConstructor();
d) None of the above
Answer: b

How many constructors can a class have?
a) Only one
b) As many as the programmer wants to define
c) Only one of each type (default, copy, etc.)
d) None of the above
Answer: b

What is the purpose of a constructor?
a) To initialize the class data members
b) To allocate memory for the class object
c) To define the behavior of the class object
d) All of the above
Answer: a

Lec 25 - OVERLOADING VS. OVERRIDING

1. **What is overloading in object-oriented programming?**
   a. Creating multiple functions with the same name but different parameters
   b. Redefining a function in a subclass that was originally defined in a superclass
   c. Both a and b
   Answer: a

   What is overriding in object-oriented programming?
   a. Creating multiple functions with the same name but different parameters
   b. Redefining a function in a subclass that was originally defined in a superclass
   c. Both a and b
   Answer: b

   Can overloaded functions have the same number of parameters?
   a. Yes
   b. No
   Answer: Yes

   Can overloaded functions have the same name and parameters?
   a. Yes
   b. No
   Answer: No

   Can overridden functions have the same name and parameters?
   a. Yes
   b. No
   Answer: Yes

   Is overloading static or dynamic polymorphism?
   a. Static
   b. Dynamic
   Answer: Static

   Is overriding static or dynamic polymorphism?
   a. Static
   b. Dynamic
   Answer: Dynamic

   Can overloading be done in the same class?
   a. Yes
   b. No
   Answer: Yes

   Can overriding be done in the same class?
   a. Yes
   b. No
   Answer: No

   What happens if an overridden function is called on an object of the subclass?
   a. The function defined in the superclass is called

b. The function defined in the subclass is called
c. Both functions are called
Answer: b

Lec 26 - BASE INITIALIZATION

1. **What is base initialization in object-oriented programming?**
   a. A mechanism to initialize the derived class data members before the base class constructor is called
   b. A mechanism to initialize the base class data members before the derived class constructor is called
   c. A mechanism to initialize both the base class and derived class data members together
   d. A mechanism to dynamically allocate memory for the base class and derived class data members
   Answer: b. A mechanism to initialize the base class data members before the derived class constructor is called

   Why is base initialization important in C++?
   a. It ensures that the derived class data members are initialized properly
   b. It avoids unnecessary default constructor calls
   c. It allows the programmer to explicitly call the constructor of the base class
   d. All of the above
   Answer: d. All of the above

   Where is the base initialization list typically located in a C++ constructor?
   a. At the beginning of the constructor body
   b. After the constructor body
   c. Before the constructor declaration
   d. None of the above
   Answer: a. At the beginning of the constructor body

   Which of the following is an advantage of using base initialization?
   a. It can improve performance by avoiding unnecessary default constructor calls
   b. It can make code more readable and maintainable
   c. It can ensure that base class data members are initialized properly
   d. All of the above
   Answer: d. All of the above

   What is the syntax for using base initialization in C++?
   a. baseClassName(argumentList), constructorBody
   b. constructorBody, baseClassName(argumentList)
   c. baseClassName(argumentList) : constructorBody
   d. None of the above
   Answer: c. baseClassName(argumentList) : constructorBody

   When should base initialization be used in C++?
   a. When the base class has a parameterized constructor
   b. When the derived class needs to initialize data members that are dependent on the values of the base class data members
   c. When the base class has const data members that cannot be initialized in the derived class constructor
   d. All of the above
   Answer: d. All of the above

   Which of the following is an example of base initialization in C++?
   a. A derived class constructor that initializes the base class data members using default values

b. A derived class constructor that initializes the base class data members using constructor arguments
c. A derived class constructor that initializes the derived class data members using constructor arguments
d. None of the above
Answer: b. A derived class constructor that initializes the base class data members using constructor arguments

Can base initialization be used to initialize data members of both the base class and derived class?
a. Yes, it can be used to initialize both the base class and derived class data members
b. No, it can only be used to initialize the base class data members
c. It depends on the constructor arguments passed in
d. None of the above
Answer: b. No, it can only be used to initialize the base class data members

Which of the following is an example of a scenario where base initialization is useful?
a. When a derived class needs to allocate memory for the base class data members
b. When a derived class needs to call the default constructor of the base class
c. When a derived class needs to initialize const data members in the base class
d. None of the above
Answer: c. When a derived class needs to initialize const data members in the base class

What is the difference between base initialization and default initialization in C++?
a. Base initialization initializes the base class data members, while default initialization initializes the derived class data members

Lec 27 - SPECIALIZATION (RESTRICTION)

1. **Which of the following is a restriction of specialization in C++?**
   A) You can partially specialize function templates
   B) You can specialize function templates for built-in types
   C) You can specialize function templates for any type
   D) Specialization can lead to code duplication and maintenance issues
   Answer: B

   What is specialization in C++?
   A) A mechanism that allows programmers to define a different implementation of a template or function for a specific set of arguments
   B) A way to restrict access to certain parts of a program
   C) A technique used to improve the performance of a program
   D) None of the above
   Answer: A

   Which of the following is not a restriction of specialization in C++?
   A) You cannot partially specialize function templates
   B) You cannot specialize function templates for built-in types
   C) You cannot specialize function templates for any type
   D) None of the above
   Answer: D

   When is specialization useful in C++?
   A) When the default behavior of a template or function is not suitable for a particular data type or value
   B) When you want to restrict access to certain parts of a program
   C) When you want to improve the performance of a program
   D) None of the above
   Answer: A

   Can you partially specialize function templates in C++?
   A) Yes
   B) No
   Answer: B

   Can you specialize function templates for a built-in type such as int or double in C++?
   A) Yes
   B) No
   Answer: B

   What are some restrictions of specialization in C++?
   A) You can partially specialize function templates
   B) You can specialize function templates for any type
   C) Specialization can lead to code duplication and maintenance issues
   D) None of the above
   Answer: C

   How does specialization help in C++?
   A) By allowing programmers to define a different implementation of a template or function for a

specific set of arguments
B) By restricting access to certain parts of a program
C) By improving the performance of a program
D) None of the above
Answer: A

What is the syntax for specialization in C++?
A) template <> function_name<>(){}
B) template <> function_name<>{}()
C) template <typename T> function_name<T>(){}
D) None of the above
Answer: A

Is overuse of specialization a good practice in C++?
A) Yes
B) No
Answer: B

Lec 28 - VIRTUAL FUNCTIONS

1. **What is the purpose of virtual functions in C++?**
   A. To achieve static polymorphism
   B. To achieve dynamic polymorphism
   C. To improve code readability
   D. To increase program performance
   Answer: B

   Which keyword is used to declare a function as virtual in C++?
   A. static
   B. virtual
   C. dynamic
   D. polymorphic
   Answer: B

   In which class are virtual functions declared in C++?
   A. Base class
   B. Derived class
   C. Abstract class
   D. Static class
   Answer: A

   Which function is called when a virtual function is invoked through a base class pointer?
   A. Base class function
   B. Derived class function
   C. Default function
   D. Static function
   Answer: B

   What is a virtual function table (vtable) in C++?
   A. A table that stores the addresses of all virtual functions in a class hierarchy
   B. A table that stores the names of all virtual functions in a class hierarchy
   C. A table that stores the values of all virtual functions in a class hierarchy
   D. A table that stores the types of all virtual functions in a class hierarchy
   Answer: A

   Can a derived class override a non-virtual function of its base class in C++?
   A. Yes
   B. No
   Answer: A

   What is the syntax for providing a default implementation of a virtual function in C++?
   A. virtual void functionName() { ... }
   B. virtual void functionName() = 0;
   C. virtual void functionName() default;
   D. virtual void functionName() { ... } default;
   Answer: D

   What is the difference between a pure virtual function and a virtual function with a default

implementation in C++?
A. A pure virtual function has no implementation, while a virtual function with a default implementation does
B. A pure virtual function cannot be called, while a virtual function with a default implementation can be
C. A pure virtual function is declared with the = 0 syntax, while a virtual function with a default implementation is declared with the = default syntax
D. There is no difference between the two
Answer: A

Can virtual functions be defined as private in a C++ class?
A. Yes
B. No
Answer: A

What is the purpose of a virtual destructor in C++?
A. To improve program performance
B. To allow objects to be destroyed properly in a class hierarchy
C. To prevent memory leaks
D. To allow objects to be cloned easily
Answer: B

Lec 29 - ABSTRACT CLASSES

1. **Which keyword is used to define an abstract class in C++?**
   a) virtual
   b) abstract
   c) interface
   d) class
   Answer: b) abstract

   Which of the following is true about abstract classes?
   a) They can be instantiated
   b) They cannot be inherited
   c) They provide a common interface for derived classes
   d) They do not allow any data members
   Answer: c) They provide a common interface for derived classes

   Which of the following is true about pure virtual functions?
   a) They have an implementation in the base class
   b) They can be called from the base class
   c) They must be overridden in the derived class
   d) They cannot be overridden in the derived class
   Answer: c) They must be overridden in the derived class

   Can an abstract class have concrete (non-virtual) functions?
   a) Yes
   b) No
   Answer: a) Yes

   Which of the following is a correct syntax for declaring a pure virtual function?
   a) virtual void func() const = 0;
   b) pure virtual void func() = 0;
   c) void virtual func() = 0;
   d) void func() const = 0;
   Answer: a) virtual void func() const = 0;

   Which of the following is a correct way to create an instance of an abstract class?
   a) Shape s;
   b) Shape* s = new Shape();
   c) Circle c;
   d) None of the above
   Answer: d) None of the above

   Which of the following is true about abstract classes and interfaces?
   a) They are the same thing
   b) Interfaces cannot have data members
   c) Abstract classes cannot have pure virtual functions
   d) None of the above
   Answer: b) Interfaces cannot have data members

   Which of the following is an advantage of using abstract classes?
   a) They allow multiple inheritance

b) They allow for runtime polymorphism
c) They provide a mechanism for code reuse
d) They can be instantiated
Answer: c) They provide a mechanism for code reuse

Which of the following is not an example of an abstract class?
a) Shape
b) Animal
c) Car
d) Vehicle
Answer: c) Car

Which of the following statements is true about abstract classes?
a) All member functions must be pure virtual functions
b) Abstract classes cannot have constructors
c) Abstract classes cannot have concrete functions
d) Abstract classes cannot have data members
Answer: c) Abstract classes cannot have concrete functions

Lec 30 - POLYMORPHISM – CASE STUDY: A SIMPLE PAYROLL APPLICATION

1. **What is polymorphism?**
   A) The ability of an object to take on many forms
   B) The ability of an object to change its type at runtime
   C) The ability of an object to inherit multiple classes
   D) The ability of an object to override its superclass methods
   Answer: A

   What is a common use case for polymorphism?
   A) Implementing different payment methods
   B) Creating complex mathematical algorithms
   C) Defining new data types
   D) All of the above
   Answer: A

   In a payroll application, how might polymorphism be used?
   A) To implement a common interface for calculating employee pay
   B) To ensure that all employees are paid the same amount
   C) To limit the types of employees that can be added to the system
   D) To prevent employees from accessing each other's pay information
   Answer: A

   What is an interface in Java?
   A) A concrete implementation of a class
   B) A template for defining a class
   C) A set of methods and constants that a class must implement
   D) A way to declare private methods in a class
   Answer: C

   What is method overloading?
   A) Defining multiple methods with the same name but different parameters
   B) Defining multiple methods with the same name and same parameters
   C) Overriding a superclass method with a subclass method
   D) None of the above
   Answer: A

   What is method overriding?
   A) Defining multiple methods with the same name but different parameters
   B) Defining multiple methods with the same name and same parameters
   C) Overriding a superclass method with a subclass method
   D) None of the above
   Answer: C

   What is the difference between method overloading and method overriding?
   A) Method overloading is done at compile time, while method overriding is done at runtime
   B) Method overloading is done by the superclass, while method overriding is done by the subclass
   C) Method overloading is used to define new methods, while method overriding is used to modify existing methods

D) Method overloading is based on the number and type of parameters, while method overriding is based on the method name and parameters
Answer: D

Can a subclass access private methods and fields of its superclass?
A) Yes, always
B) No, never
C) Only if the subclass is in the same package as the superclass
D) Only if the superclass declares the methods and fields as protected
Answer: B

What is an abstract class in Java?
A) A class that cannot be instantiated
B) A class that does not have any methods
C) A class that only has private methods
D) A class that can only be used as a superclass
Answer: A

Can an abstract class have non-abstract methods?
A) Yes, but only if the class also has at least one abstract method
B) Yes, but only if the class has no abstract methods
C) No, an abstract class can only have abstract methods
D) None of the above
Answer: B

Lec 31 - MULTIPLE INHERITANCE

1. **What is multiple inheritance in object-oriented programming?**
   A) Inheriting from multiple subclasses
   B) Inheriting from multiple parent classes
   C) Inheriting from multiple sibling classes
   D) Inheriting from multiple child classes
   Answer: B

   Which programming languages support multiple inheritance?
   A) Java
   B) Python
   C) C++
   D) All of the above
   Answer: D

   What is the main advantage of multiple inheritance?
   A) Code reusability
   B) Improved modularity
   C) Enhanced flexibility
   D) Reduced complexity
   Answer: C

   What is the potential downside of using multiple inheritance?
   A) Code duplication
   B) Ambiguity in method and attribute resolution
   C) Inability to access parent class attributes and methods
   D) Reduced code readability
   Answer: B

   What is the diamond problem in the context of multiple inheritance?
   A) A conflict that arises when two classes define the same attribute or method
   B) A conflict that arises when two classes inherit from the same parent class
   C) A conflict that arises when a class inherits from two or more classes that have a common ancestor
   D) A conflict that arises when a class has multiple methods with the same name and arguments
   Answer: C

   How can the diamond problem be resolved?
   A) By renaming conflicting methods and attributes
   B) By removing one of the conflicting parent classes
   C) By using virtual inheritance
   D) By using multiple inheritance only when necessary
   Answer: C

   What is the syntax for implementing multiple inheritance in Python?
   A) class ChildClass(ParentClass1, ParentClass2):
   B) class ChildClass(ParentClass1 and ParentClass2):
   C) class ChildClass(ParentClass1 or ParentClass2):
   D) class ChildClass(ParentClass1 ParentClass2):
   Answer: A

   What is the order of method resolution in multiple inheritance?

B) Breadth-first search
C) Random order
D) Alphabetical order
Answer: A

What is the role of the super() function in multiple inheritance?
A) It calls the constructor of the parent class
B) It calls the method of the parent class
C) It resolves conflicts in method and attribute names
D) It prevents ambiguity in method resolution
Answer: B

What is the difference between multiple inheritance and interface inheritance?
A) Multiple inheritance allows a class to inherit from multiple classes, while interface inheritance allows a class to inherit only method signatures.
B) Multiple inheritance allows a class to inherit only method signatures, while interface inheritance allows a class to inherit from multiple classes.
C) Multiple inheritance and interface inheritance are the same thing.
D) Multiple inheritance and interface inheritance are not related concepts.
Answer: A

Lec 32 - GENERIC PROGRAMMING

1. **What is the purpose of generic programming?**
   a) To improve code readability
   b) To increase code efficiency
   c) To write reusable code
   d) To decrease code maintainability

   Answer: c) To write reusable code

   Which programming paradigm is most commonly associated with generic programming?
   a) Object-oriented programming
   b) Procedural programming
   c) Functional programming
   d) Event-driven programming

   Answer: a) Object-oriented programming

   In C++, what is the primary mechanism for achieving generic programming?
   a) Templates
   b) Polymorphism
   c) Inheritance
   d) Encapsulation

   Answer: a) Templates

   What is the advantage of using templates in C++?
   a) Templates reduce code complexity and improve code readability
   b) Templates allow for more efficient code execution
   c) Templates enable code to be reused with different data types
   d) Templates make it easier to write object-oriented code

   Answer: c) Templates enable code to be reused with different data types

   In Java, what is the primary mechanism for achieving generic programming?
   a) Templates
   b) Polymorphism
   c) Inheritance
   d) Generics

   Answer: d) Generics

   What is the difference between templates in C++ and generics in Java?
   a) Templates are more efficient than generics
   b) Templates are a more powerful mechanism for achieving generic programming than generics
   c) Templates are more difficult to use than generics
   d) Templates require explicit type parameterization, while generics do not

   Answer: d) Templates require explicit type parameterization, while generics do not

   Which of the following is an example of a generic algorithm?
   a) Bubble sort

b) Quick sort
c) Binary search
d) All of the above

Answer: d) All of the above

Which of the following is an advantage of generic algorithms?
a) Generic algorithms are more efficient than non-generic algorithms
b) Generic algorithms can be used with any data type
c) Generic algorithms can only be used with primitive data types
d) Generic algorithms are easier to debug than non-generic algorithms

Answer: b) Generic algorithms can be used with any data type

Which of the following is a disadvantage of using generics in Java?
a) Generics can lead to code bloat
b) Generics can be slower than non-generic code
c) Generics can make code harder to read
d) Generics can lead to type erasure

Answer: d) Generics can lead to type erasure

Which of the following is an example of a generic class in C++?
a) std::vector
b) std::map
c) std::pair
d) All of the above

Answer: d) All of the above

Lec 33 - MULTIPLE TYPE ARGUMENTS

1. **What are multiple type arguments?**
   a) A type of argument that can only be used with a specific data type
   b) The ability to define multiple data types for use with a generic class or function
   c) A type of argument that is only used in functional programming
   Answer: b

   Which programming languages support multiple type arguments?
   a) C++
   b) Java
   c) Python
   d) All of the above
   Answer: d

   What is the advantage of using multiple type arguments?
   a) Increased flexibility in the use of generic programming
   b) Reduced code complexity
   c) Improved code efficiency
   Answer: a

   Which keyword is used to define multiple type arguments in Java?
   a) class
   b) template
   c) < >
   Answer: c

   Which programming paradigm uses multiple type arguments extensively?
   a) Object-oriented programming
   b) Functional programming
   c) Imperative programming
   Answer: a

   Can multiple type arguments be used with functions in C++?
   a) Yes
   b) No
   Answer: a

   What is the syntax for defining multiple type arguments in C++?
   a) template<typename T, U>
   b) template<class T, class U>
   c) template<class T, U>
   Answer: b

   What is the default type argument in Java?
   a) Object
   b) Integer
   c) Double
   Answer: a

   How many type arguments can be defined for a generic class in C++?
   a) One

b) Two
c) Any number
Answer: c

What is the difference between single type arguments and multiple type arguments?
a) Single type arguments can only be used with one data type, while multiple type arguments can be used with multiple data types
b) Single type arguments are used in functional programming, while multiple type arguments are used in object-oriented programming
c) There is no difference between single type arguments and multiple type arguments
Answer: a

Lec 34 - GENERIC ALGORITHMS

1. **What are generic algorithms in programming?**
   a. Algorithms that work with a specific data type only
   b. Algorithms that work with any data type
   c. Algorithms that are optimized for performance
   Answer: b

   In which programming paradigm are generic algorithms commonly used?
   a. Object-oriented programming
   b. Procedural programming
   c. Functional programming
   Answer: a

   What is the main advantage of using generic algorithms?
   a. Improved performance
   b. Increased code complexity
   c. Reusability and adaptability of code
   Answer: c

   Which programming languages support generic algorithms?
   a. C++
   b. Java
   c. Python
   d. All of the above
   Answer: d

   Can generic algorithms be used with user-defined data types?
   a. Yes
   b. No
   Answer: a

   What is the syntax for using generic algorithms in C++?
   a. < >
   b. { }
   c. ( )
   Answer: a

   Which standard library in C++ provides support for generic algorithms?
   a. stdio.h
   b. iostream
   c. algorithm
   Answer: c

   What is the purpose of the std::sort algorithm in C++?
   a. To sort elements in ascending order
   b. To sort elements in descending order
   c. To remove duplicate elements
   Answer: a

   Which of the following is an example of a generic algorithm?
   a. Bubble sort

b. Quick sort
c. Binary search
Answer: c

What is the main disadvantage of using generic algorithms?
a. Limited applicability to specific data types
b. Reduced performance compared to specialized algorithms
c. Increased code complexity
Answer: b

Lec 35 - MEMBER TEMPLATES

1. **What are member templates in C++?**
   A. Templates defined inside a class or struct
   B. Templates defined outside a class or struct
   C. Templates that cannot be used with any data type
   D. Templates that can only be used with integer data types
   Answer: A. Templates defined inside a class or struct.

   What is the advantage of using member templates?
   A. Increased reusability and adaptability of code
   B. Improved code performance
   C. Reduced code complexity
   D. All of the above
   Answer: D. All of the above.

   Can member templates access the data members and methods of the class they are defined in?
   A. Yes
   B. No
   Answer: A. Yes.

   Can member templates be used to provide generic constructors?
   A. Yes
   B. No
   Answer: A. Yes.

   What is the syntax for defining a member template?
   A. template<class T> void myFunction(T arg);
   B. template<typename T> void myFunction(T arg);
   C. template<class T> struct MyClass {...};
   D. All of the above
   Answer: D. All of the above.

   Can member templates be specialized for specific data types?
   A. Yes
   B. No
   Answer: A. Yes.

   What is the purpose of a member function template?
   A. To provide a generic member function that can work with any data type
   B. To provide a specialized member function for a specific data type
   C. To provide a constructor for a class or struct
   D. None of the above
   Answer: A. To provide a generic member function that can work with any data type.

   What is the advantage of using a member function template over a regular member function?
   A. Increased reusability and adaptability of code
   B. Improved code performance

C. Reduced code complexity
D. All of the above
Answer: D. All of the above.

What is the difference between a member function template and a regular member function?
A. A member function template can work with any data type, whereas a regular member function can only work with specific data types
B. A member function template is defined inside a class or struct, whereas a regular member function is defined outside the class or struct
C. A member function template cannot access the data members and methods of the class it is defined in, whereas a regular member function can
D. None of the above
Answer: A. A member function template can work with any data type, whereas a regular member function can only work with specific data types.

What is the purpose of template specialization?
A. To provide a generic template that can work with any data type
B. To provide a specialized template for a specific data type
C. To provide a constructor for a class or struct
D. None of the above
Answer: B. To provide a specialized template for a specific data type.

Lec 36 - MEMBER TEMPLATES REVISITED

1. **What is member templates revisited in C++?**
   a) A way to create specialized versions of member function templates
   b) A technique to simplify the syntax of member function templates
   c) A type of member function template that only works with specific data types
   d) A way to access private data members of a class using member templates
   Answer: b) A technique to simplify the syntax of member function templates

   What is the benefit of using member templates revisited?
   a) Reduced code duplication
   b) Improved performance
   c) Simplified syntax
   d) All of the above
   Answer: d) All of the above

   What is template argument deduction?
   a) The process of providing explicit template arguments to a function template
   b) The process of inferring template arguments from function arguments
   c) The process of specializing a template for a specific data type
   d) The process of defining a template inside a class
   Answer: b) The process of inferring template arguments from function arguments

   What is the syntax for using member templates revisited?
   a) auto func(args...)
   b) template auto func(args...)
   c) template<typename T> auto Class::func(T arg)
   d) template<typename T> auto Class<T>::func(args...)
   Answer: d) template<typename T> auto Class<T>::func(args...)

   Can member templates revisited be used with constructors?
   a) Yes
   b) No
   Answer: a) Yes

   What is the difference between regular member function templates and member function templates revisited?
   a) Member function templates revisited use template argument deduction
   b) Member function templates revisited can only be used with specific data types
   c) Regular member function templates have a simpler syntax
   d) Regular member function templates cannot be specialized
   Answer: a) Member function templates revisited use template argument deduction

   What is the purpose of template argument deduction in member templates revisited?
   a) To simplify the syntax of member function templates
   b) To reduce code duplication
   c) To allow for specialization of member function templates
   d) To infer the template arguments from the function arguments
   Answer: d) To infer the template arguments from the function arguments

   Can member templates revisited be used with non-static member functions?
   a) Yes

b) No
Answer: a) Yes

What is the advantage of using member templates revisited over regular member function templates?
a) Reduced code duplication
b) Improved performance
c) More concise and readable code
d) All of the above
Answer: d) All of the above

What is the disadvantage of using member templates revisited?
a) Limited support for certain compilers
b) Increased complexity
c) Slower compile times
d) None of the above
Answer: d) None of the above

Lec 37 - RESOLUTION ORDER

1. **What is resolution order in programming?**
   a) The order in which legal disputes are resolved
   b) The order in which software applications resolve conflicts
   c) The order in which functions are defined
   d) The order in which variables are declared
   Answer: b

   Which of the following statements is true about resolution order?
   a) It is only relevant in legal disputes
   b) It determines the order in which variables are initialized
   c) It can be defined by explicit rules or by default
   d) It is only used in object-oriented programming
   Answer: c

   What is the default resolution order for Python classes?
   a) Depth-first search
   b) Breadth-first search
   c) Left-to-right
   d) Right-to-left
   Answer: a

   In Java, which keyword is used to explicitly specify the resolution order for method calls?
   a) super
   b) this
   c) extends
   d) implements
   Answer: a

   In C++, what is the resolution order for overloaded operators?
   a) Left-to-right
   b) Right-to-left
   c) Undefined
   d) Depends on the operator being overloaded
   Answer: c

   What is the resolution order for CSS styles?
   a) Inline styles, internal styles, external styles
   b) External styles, internal styles, inline styles
   c) Internal styles, inline styles, external styles
   d) Inline styles, external styles, internal styles
   Answer: d

   In Perl, which operator is used to explicitly specify the resolution order for method calls?
   a) ->>
   b) <<
   c) ::
   d) //
   Answer: c

   What is the resolution order for conflicting domain name records?
   a) MX, CNAME, A

b) CNAME, A, MX
c) A, CNAME, MX
d) MX, A, CNAME
Answer: b

In Python, what is the resolution order for multiple inheritance?
a) Left-to-right
b) Right-to-left
c) Depth-first search
d) Breadth-first search
Answer: c

In Ruby, what is the resolution order for method calls?
a) Right-to-left
b) Left-to-right
c) Depth-first search
d) Breadth-first search
Answer: c

Lec 38 - FUNCTION TEMPLATE OVERLOADING

1. **What is function template overloading in C++?**
   A) Creating multiple functions with the same name and argument types
   B) Creating multiple functions with the same name but different argument types
   C) Creating a single function that can be used with different data types
   D) Creating a single function with multiple return types
   Answer: B

   How is function template overloading achieved in C++?
   A) By defining multiple functions with different names
   B) By defining multiple functions with different return types
   C) By defining a single function with multiple argument types
   D) By defining a function template with placeholders for argument types
   Answer: D

   What is the purpose of function template overloading in C++?
   A) To reduce the number of functions in a program
   B) To create more complex functions
   C) To improve code flexibility and reusability
   D) To improve program performance
   Answer: C

   Can function templates be overloaded based on the return type?
   A) Yes
   B) No
   Answer: B

   What is the difference between function overloading and function template overloading?
   A) Function overloading is limited to a single data type, while function template overloading allows for multiple data types.
   B) Function template overloading is limited to a single data type, while function overloading allows for multiple data types.
   C) Function overloading creates multiple functions with the same name and argument types, while function template overloading creates multiple functions with the same name but different argument types.
   D) There is no difference between function overloading and function template overloading.
   Answer: C

   Can function templates be overloaded based on the number of arguments?
   A) Yes
   B) No
   Answer: A

   Which of the following is an advantage of function template overloading?
   A) It makes the code more complex
   B) It makes the code less flexible
   C) It improves code flexibility and reusability
   D) It improves program performance
   Answer: C

   Can function templates be overloaded based on the constness of the arguments?
   A) Yes

B) No
Answer: A

Which of the following is true regarding function template overloading in C++?
A) Only one function template can be defined for a given set of argument types
B) Multiple function templates can be defined for a given set of argument types
C) Function template overloading is not supported in C++
D) Function template overloading is only supported for built-in data types
Answer: B

Can function templates be overloaded based on the type of argument?
A) Yes
B) No
Answer: A

Lec 39 - TEMPLATES & STATIC MEMBERS

1. **Which keyword is used to define a template in C++?**
   a) template
   b) class
   c) typename
   d) all of the above
   Answer: a) template

   Which of the following is not a benefit of using templates in C++?
   a) Code reusability
   b) Improved efficiency
   c) Flexibility
   d) Simplified syntax
   Answer: d) Simplified syntax

   What is the purpose of static members in C++?
   a) To create class objects
   b) To provide a single instance of a variable for all class objects
   c) To define functions that can be accessed without creating an object
   d) To provide a way to create objects dynamically
   Answer: b) To provide a single instance of a variable for all class objects

   Which keyword is used to declare a static member in a class definition?
   a) static
   b) const
   c) friend
   d) virtual
   Answer: a) static

   Which of the following is true about static member functions in C++?
   a) They can be called using an object of the class.
   b) They cannot access non-static members of the class.
   c) They can only be declared in the private section of a class.
   d) They cannot be overloaded.
   Answer: b) They cannot access non-static members of the class.

   Which of the following is not a valid template parameter type in C++?
   a) int
   b) float
   c) void
   d) char*
   Answer: b) float

   What is the purpose of the typename keyword in template definitions?
   a) To indicate a class type
   b) To indicate a function type
   c) To indicate a pointer type
   d) To indicate a void type
   Answer: a) To indicate a class type

   Can a static member of a class be accessed using the class name and the scope

resolution operator?
a) Yes
b) No
Answer: a) Yes

Can a template function be defined outside the class definition?
a) Yes
b) No
Answer: a) Yes

Which of the following is not a valid way to specialize a template function?
a) Explicit specialization
b) Partial specialization
c) Function overloading
d) None of the above
Answer: c) Function overloading

Lec 40 - CURSORS

1. **What is a cursor in a database management system?**
   A) A database object that stores data
   B) A pointer that enables traversal over a set of rows in a result set
   C) A file that stores database information
   D) A function that returns a value
   Answer: B

   What is the purpose of a cursor in SQL?
   A) To insert data into a database
   B) To retrieve data from a database
   C) To update data in a database
   D) To delete data from a database
   Answer: B

   What are the types of cursors in SQL?
   A) Static, dynamic, and keyset-driven
   B) Primary, secondary, and tertiary
   C) Logical, physical, and virtual
   D) Simple, complex, and compound
   Answer: A

   Which SQL keyword is used to define a cursor?
   A) DECLARE
   B) CREATE
   C) INSERT
   D) SELECT
   Answer: A

   What is the purpose of the OPEN statement in SQL cursors?
   A) To define the cursor
   B) To fetch the next row from the cursor
   C) To close the cursor
   D) To execute a stored procedure
   Answer: B

   Which SQL keyword is used to fetch the next row from a cursor?
   A) FETCH
   B) SELECT
   C) UPDATE
   D) DELETE
   Answer: A

   What is the purpose of the CLOSE statement in SQL cursors?
   A) To define the cursor
   B) To fetch the next row from the cursor
   C) To close the cursor
   D) To execute a stored procedure
   Answer: C

   What is the purpose of the DEALLOCATE statement in SQL cursors?
   A) To define the cursor

B) To fetch the next row from the cursor
C) To close the cursor
D) To free up memory used by the cursor
Answer: D

Which type of cursor is more efficient in terms of performance?
A) Static cursor
B) Dynamic cursor
C) Keyset-driven cursor
D) There is no difference in performance between cursor types
Answer: A

Can cursors be used in all programming languages?
A) Yes
B) No, only in SQL and related languages
C) No, only in procedural languages
D) No, only in object-oriented languages
Answer: B

Lec 41 - STANDARD TEMPLATE LIBRARY

1. **What is the purpose of the Standard Template Library (STL) in C++?**
   a) To provide a library of generic algorithms, data structures, and iterators
   b) To provide a library of graphics functions
   c) To provide a library of mathematical functions
   d) To provide a library of input/output functions
   Answer: a

   Which of the following is a container class in STL?
   a) Vector
   b) Set
   c) List
   d) All of the above
   Answer: d

   Which of the following is not a type of iterator in STL?
   a) Forward iterator
   b) Reverse iterator
   c) Bidirectional iterator
   d) Parallel iterator
   Answer: d

   What is the complexity of finding an element in a set in STL?
   a) O(1)
   b) O(log n)
   c) O(n)
   d) O(n^2)
   Answer: b

   Which algorithm in STL is used to sort a range of elements in ascending order?
   a) sort
   b) reverse
   c) merge
   d) unique
   Answer: a

   Which container class in STL stores unique elements in sorted order?
   a) Vector
   b) Map
   c) Set
   d) Queue
   Answer: c

   What is the difference between a stack and a queue in STL?
   a) A stack is a LIFO structure, while a queue is a FIFO structure.
   b) A stack is a FIFO structure, while a queue is a LIFO structure.
   c) Both are LIFO structures.
   d) Both are FIFO structures.
   Answer: a

   Which algorithm in STL is used to copy a range of elements from one container to

another?
a) find
b) copy
c) sort
d) replace
Answer: b

Which of the following is not a header file in STL?
a) <vector>
b) <set>
c) <list>
d) <iostream>
Answer: d

Which container class in STL is used to implement a priority queue?
a) Map
b) Queue
c) Stack
d) Heap
Answer: d

Lec 42 - ITERATORS

1. **Which of the following is NOT a type of iterator in C++?**
   a) Forward iterator
   b) Backward iterator
   c) Input iterator
   d) Random access iterator
   Answer: b) Backward iterator

   Which of the following is a feature of a forward iterator?
   a) Can move in both directions
   b) Can be used to modify the elements in a container
   c) Can access the elements of a container multiple times
   d) Can only move forward in a container
   Answer: d) Can only move forward in a container

   Which of the following is an example of a container that supports random access iterators?
   a) Linked list
   b) Queue
   c) Array
   d) Set
   Answer: c) Array

   What is the purpose of an output iterator?
   a) To iterate through a container in reverse order
   b) To modify the elements in a container
   c) To read the elements in a container
   d) To write data to a container
   Answer: d) To write data to a container

   Which of the following is an example of a bidirectional iterator?
   a) Stack
   b) Deque
   c) Forward list
   d) Map
   Answer: b) Deque

   What is the complexity of the operator++() function for a random access iterator?
   a) O(n)
   b) O(log n)
   c) O(1)
   d) O(n^2)
   Answer: c) O(1)

   Which of the following algorithms require a random access iterator?
   a) std::sort()
   b) std::transform()
   c) std::reverse()
   d) std::unique()
   Answer: a) std::sort()

   Which of the following is a characteristic of an input iterator?

b) Can only access the elements of a container once
c) Can move in both directions
d) Can skip elements in a container
Answer: b) Can only access the elements of a container once

Which of the following is an example of a container that supports bidirectional iterators?
a) Hash table
b) Binary search tree
c) Vector
d) Queue
Answer: b) Binary search tree

Which of the following is an example of a container that supports forward iterators?
a) Stack
b) Map
c) Linked list
d) Set
Answer: c) Linked list

Lec 43 - EXAMPLE – ABNORMAL TERMINATION

1. **What is abnormal termination?**
   A. A program that runs indefinitely
   B. A program that terminates normally
   C. A program that terminates unexpectedly due to an error or exception
   D. A program that never compiles
   Answer: C

   What can cause abnormal termination?
   A. Memory access violations
   B. Stack overflow
   C. Division by zero
   D. All of the above
   Answer: D

   What is a segmentation fault?
   A. A type of error that occurs when a program attempts to access memory that has already been freed or is outside of its allocated range
   B. A type of error that occurs when a program runs out of stack space
   C. A type of error that occurs when a program attempts to divide by zero
   D. A type of error that occurs when a program fails to compile
   Answer: A

   What is an access violation?
   A. A type of error that occurs when a program attempts to access memory that has already been freed or is outside of its allocated range
   B. A type of error that occurs when a program runs out of stack space
   C. A type of error that occurs when a program attempts to divide by zero
   D. A type of error that occurs when a program fails to compile
   Answer: A

   What is a floating-point exception?
   A. A type of error that occurs when a program attempts to divide a number by zero
   B. A type of error that occurs when a program attempts to access memory that has already been freed or is outside of its allocated range
   C. A type of error that occurs when a program runs out of stack space
   D. A type of error that occurs when a program fails to compile
   Answer: A

   What is integer division by zero?
   A. A type of error that occurs when a program attempts to divide an integer by zero
   B. A type of error that occurs when a program attempts to access memory that has already been freed or is outside of its allocated range
   C. A type of error that occurs when a program runs out of stack space
   D. A type of error that occurs when a program fails to compile
   Answer: A

   How can abnormal termination be prevented?
   A. Implementing error handling and exception handling mechanisms

B. Proper testing and debugging practices
C. Both A and B
D. None of the above
Answer: C

What is error handling?
A. A mechanism to catch and handle errors or exceptions
B. A way to prevent programs from crashing
C. A way to write more efficient code
D. A way to increase program security
Answer: A

What is exception handling?
A. A mechanism to catch and handle errors or exceptions
B. A way to prevent programs from crashing
C. A way to write more efficient code
D. A way to increase program security
Answer: A

What is debugging?
A. The process of identifying and fixing errors in a program
B. The process of optimizing a program for performance
C. The process of writing code
D. The process of designing a program
Answer: A

Lec 44 - STACK UNWINDING

1. **What is stack unwinding?**
   a) A process of releasing memory from the heap
   b) A process of releasing memory from the stack
   c) A process of clearing the call stack in response to an exception being thrown
   d) A process of allocating memory on the stack
   Answer: c

   What happens during stack unwinding?
   a) All functions that were called before the exception occurred are popped off the stack
   b) All functions that were called after the exception occurred are popped off the stack
   c) All functions in the program are popped off the stack
   d) All variables in the program are popped off the stack
   Answer: a

   Why is stack unwinding important?
   a) It prevents resource leaks in C++ programs
   b) It allows programs to allocate memory on the stack
   c) It improves program performance
   d) It allows programs to release memory from the heap
   Answer: a

   What is the purpose of calling destructors during stack unwinding?
   a) To release any dynamically allocated memory or resources
   b) To allocate memory on the stack
   c) To improve program performance
   d) To initialize variables on the stack
   Answer: a

   What happens if an exception is thrown but not caught?
   a) The program terminates immediately
   b) The program continues to execute normally
   c) The program enters an infinite loop
   d) The program enters a state of undefined behavior
   Answer: a

   What is the role of the try block in stack unwinding?
   a) It contains the code that may throw an exception
   b) It contains the code that is executed if an exception is caught
   c) It contains the code that is executed if an exception is not caught
   d) It contains the code that is executed before stack unwinding begins
   Answer: a

   What is the role of the catch block in stack unwinding?
   a) It catches and handles exceptions
   b) It releases memory from the heap
   c) It initializes variables on the stack
   d) It allocates memory on the stack
   Answer: a

   What happens if a function throws an exception but does not have a catch block?
   a) The program terminates immediately

b) The program continues to execute normally
c) The exception is caught by a catch block in a higher function on the call stack
d) The program enters a state of undefined behavior
Answer: c

When are destructors called during stack unwinding?
a) In reverse order of construction
b) In the order of construction
c) Randomly
d) It depends on the implementation
Answer: a

What is the difference between stack unwinding and stack overflow?
a) Stack unwinding is intentional, while stack overflow is unintentional
b) Stack unwinding releases memory from the stack, while stack overflow overwrites memory on the stack
c) Stack unwinding occurs during exception handling, while stack overflow occurs when the stack becomes too full
d) Stack unwinding only occurs in C++, while stack overflow can occur in any programming language
Answer: c

Lec 45 - RESOURCE MANAGEMENT

1. **Which of the following is not a common system resource that requires efficient management?**
   a. Memory
   b. CPU cycles
   c. Input devices
   d. Network connections
   Answer: c

   Which of the following is not a technique used for efficient resource management?
   a. Resource pooling
   b. Resource caching
   c. Resource sharing
   d. Resource wasting
   Answer: d

   What is resource pooling?
   a. The process of allocating resources to specific tasks
   b. The process of sharing resources among multiple tasks
   c. The process of caching resources for future use
   d. The process of deallocating unused resources
   Answer: b

   Which of the following is an example of a resource leak?
   a. Using a caching mechanism for database queries
   b. Releasing memory after it has been used
   c. Failing to close a database connection
   d. Allocating resources dynamically
   Answer: c

   What is garbage collection?
   a. The process of deallocating unused memory
   b. The process of deallocating unused network connections
   c. The process of deallocating unused file handles
   d. The process of deallocating unused CPU cycles
   Answer: a

   What is resource caching?
   a. The process of allocating resources to specific tasks
   b. The process of sharing resources among multiple tasks
   c. The process of caching resources for future use
   d. The process of deallocating unused resources
   Answer: c

   What is resource sharing?
   a. The process of allocating resources to specific tasks
   b. The process of sharing resources among multiple tasks
   c. The process of caching resources for future use
   d. The process of deallocating unused resources
   Answer: b

   Which of the following is not a benefit of efficient resource management?

b. Increased reliability
c. Reduced resource usage
d. Increased memory leaks
Answer: d

Which of the following is an example of resource pooling?
a. Sharing a database connection among multiple threads
b. Allocating a fixed amount of memory for a process
c. Caching query results for future use
d. Deallocating memory when it is no longer needed
Answer: a

What is resource caching used for?
a. To allocate resources to specific tasks
b. To share resources among multiple tasks
c. To cache resources for future use
d. To deallocate unused resources
Answer: c