Teaching Guidelines for
C++ Programming
PG-DAC September 2023

Duration: 72 hours (32 theory hours + 32 lab hours + 8 revision/practice hours)

Objective: To learn object oriented programming using C++

Prerequisites: Knowledge of computer fundamentals

Evaluation: 100 marks
Weightage: CCEE – 40%, Lab exam – 40%, Internals – 20%

Text Book:
- C++ Primer Plus by Stephen Prata /Pearson

References:
- Thinking in C++ by Bruce Eckel
- The C++ Programming Language, Bjarne Stroustrup

(Note: Each Session is of 2 hours)

Sessions 1: Getting Started
Lecture:
- Installation and Setup development environment
- The need of C++
- Features of C++
- C++ versus C
- History of C++
- Writing your first C++ program

Lab:
Write different C++ programs to
- Print Hello World
- Add two numbers/binary numbers/characters
- Calculate compound interest
- Calculate power of a number
- Swap two numbers
- Calculate area of rectangle

Session 2: Beginning with C++
Lecture:
- C++Program structure
- Introduction of advanced C++ concepts and feature of C++ 17
- C++ Tokens
- Initialization
- Static Members
- Constant Members
- Expressions

Operators
- Arithmetic Operator

- Relational Operator
- Logical Operator
- Unary Operator
- Ternary Operator
- Assignment Operator

Lab:

- Write a Student class and use it in your program. Store the data of 10 students and display the sorted data according to their roll numbers, dates of birth, and total marks.
- Implement all C++ operators
- Declare members and implement in your programs.

Session 3: Conditional and Looping Statements
Lecture:

- If, else if, switch
- for loop
- while loop
- do while loop
- Jump statement (break, continue& return keyword)
- Arrays
- Declaration and initialization of an array
- 1-D and 2-D arrays

Lab:

- Implement all control structures through your program
- Implement a program which accepts command line arguments from main function.

Session 4: Functions in C++
Lecture

- Different forms of functions
- Function prototyping
- Call by Reference
- Inline Functions
- Math library functions etc.

Lab:

- Implement functions through your program
- Declare function and call it by reference and note the observations
- Implement Inline functions in your program

Sessions 5 & 6: Memory Management and Pointers
Lecture

- Introduction to memory management in C++
- Pointers in C++
- Arrays using pointers
- Enumeration
- Typedef
- Using New operator
- Class pointer
- this pointer
- Comparison of new over malloc, calloc and realloc, etc.
- Memory freeing using Delete operator

Lab:

- Assignments using pointers, arrays of pointers

- Assignments on passing pointers in functions
- Using pointers, write your own functions for the following:
    - String comparison
    - String concatenate
    - String copy
    - String length

Note: Do not include <string.h> in your program and implement Delete operator in your program.

Session 7: OOP Concepts
Lecture
- Discussion on object oriented concepts
- Classes and Objects, Access Specifiers, Overloading, Inheritance, Polymorphism
- Constructors and Destructors
- Namespaces

Lab:
- Write a student class and use it in your program. Store the data of 10 students and display the sorted data according to their roll numbers, dates of birth, and total marks.

Session 8: Constructor and Destructor
Lecture
- Constructors
- Parameterized constructors
- Multiple constructors in class
- Dynamic initialization of objects
- Copy Constructors
- Destructors

Lab:
- Implement constructor and destructors through your program
- Write a program to implement inner class in C++

Session 9: Inheritance – extending class
Lecture
- Types of inheritance
- Single inheritance
- Multiple inheritance
- Multilevel inheritance
- Hierarchical inheritance
- Hybrid inheritance, etc.
- Virtual base class
- Constructors in derived class

Lab:
- Design a hierarchy of computer printers. Use multiple inheritance in your hierarchy. Also use friend functions and classes in your program.

Session 10: Polymorphism
Lecture
- Types of Polymorphism
- Overloading functions
- Overloading Operators
- Friend functions
- Constant functions

Lab:

- Write Date and Time classes that allow you to add, subtract, read and print simple dates in dd/mm/yyyy and time in hh:mm:ss formats. Use function overloading in your program.
- Assignments to overload =, ==, +, ++, --, <<, >> and [ ]operators.

Session 11: Virtual Functions and Abstract Class
Lecture

- Run Time Polymorphism
- Virtual Functions and Pure virtual functions
- dynamic_cast, static_cast,const_cast, reinterpret_cast
- Interfaces
- Abstract class

Lab:

- Implement Abstract classes in your program
- Using virtual and pure virtual functions implement hierarchy of computer printers
- Implement diamond problem with real life example

Session 12: Exception Handling
Lecture

- Exception Handling Introduction
- Exception handling – throwing, catching, re-throwing an exception
- Specifying exceptions etc.

Lab:

- Implement exceptions and do proper management through your program
- Implement Custom exception class

Session 13: Managing Console I/O Operations
Lecture

- Introduction
- C++ streams
- C++ stream classes
- Unformatted I/O operations
- Formatted I/O operations
- Managing output with manipulators

Lab:

- Implement console I/O operations through your program.

Session 14: File Handling in C++
Lecture

- Definition of file
- File handling in C++
- Doing read, write operation in files

Lab:

- Assignments on files doing different operations

Session 15: Templates
Lecture

- Introduction to Templates
- Function Templates
- Class Templates

Lab:

- Assignments on templates

Session 16: STL and RTTI

Lecture

- Introduction to C++ Standard Library
- Introduction to RTTI (Run-Time Type Information) in C++

Lab:

- Assignments on STL Library

Teaching Guidelines for

Concepts of Operating Systems & Software Development Methodologies

PG-DAC September
2023

Duration: 72 hours (50 theory hours + 22 lab hours)

Evaluation: 100 marks
Weightage: Theory exam – 40%, Lab exam – 30%, Internals – 30%

Concepts of Operating Systems

Duration: 26 hours (18 theory hours + 8 lab hours)

Objective: To introduce Operating System concepts with Linux environment, and to learn Shell Programming.

Prerequisites: Knowledge of computer fundamentals

Evaluation: 35 marks (CCEE: 15 + Lab exam: 10 + Internals: 10)

Text Books:
- Operating Systems Principles by Abraham Silberschatz, Peter Galvin& Greg Gagne / Wiley
- Unix Concepts and Applications by Sumitabha Das / McGraw Hill

References:
- Modern operating Systems by Andrew Tanenbaum & Herbert Bos/ Pearson
- Principles of Operating Systems by Naresh Chauhan / Oxford University Press
- Beginning Linux Programming by Neil Matthew & Richard Stones / Wrox
- Operating System : A Design-Oriented Approach by Charles Crowley / McGraw Hill

---

(Note: Each Session is of 2 hours)

Session 1: Introduction to OS
Lecture:
- What is OS; How is it different from other application software; Why is it hardware dependent?
- Different components of OS
- Basic computer organization required for OS.
- Examples of well-known OS including mobile OS, embedded system OS, Real Time OS, desktop OS server machine OS etc. ; How are these different from each other and why
- Functions of OS
- User and Kernel space and mode; Interrupts and system calls

No Lab

Session 2: Introduction to Linux
Lecture:
- Working basics of file system
- Commands associated with files/directories & other basic commands. Operators like redirection, pipe
- What are file permissions and how to set them?

- Permissions (chmod, chown, etc); access control list; network commands (telenet, ftp, ssh, sftp, finger)
- System variables like – PS1, PS2 etc. How to set them

Shell Programming

- What is shell; What are different shells in Linux?
- Shell variables; Wildcard symbols
- Shell meta characters; Command line arguments; Read, Echo

Lab: (4 hours)

- Working with various OS commands
- Shell programs related to Session 2

Session 3: Shell Programming

Lecture:

- Decision loops (if else, test, nested if else, case controls, while…until, for)
- Regular expressions; Arithmetic expressions
- More examples in Shell Programming

Lab: (4 hours)

- Shell Programs related to Session 3

Sessions 4 & 5: Processes

Lecture:

- What is process; preemptive and non-preemptive processes
- Process management; Process life cycle
- What are schedulers – Short term, Mediumterm and Long term.
- Process scheduling algorithms – FCFS, Shortest Job First, Priority, RR, Queue. Belady’s Anomaly
- Examples associated with scheduling algorithms to find turnaround time to find the better performing scheduler.
- Process creation using fork; waitpid and exec system calls; Examples on process creation; Parent and child processes
- Orphan and zombie processes

No Lab

Sessions 6 & 7:

Lecture:

Memory Management

- What are different types of memories; What is the need of Memory management
- Continuous and Dynamic allocation
- First Fit, Best Fit, worst Fit
- Compaction
- Internal and external fragmentation
- Segmentation – What is segmentation; Hardware requirement for segmentation; segmentation table and its interpretation
- Paging – What is paging; hardware required for paging; paging table; Translation look aside buffer
- Concept of dirty bit
- Shared pages and reentrant code
- Throttling

No Lab

Session 8:

Lecture:

Virtual Memory

- What is virtual memory
- Demand paging
- Page faults
- Page replacement algorithms

No Lab

Session 9:

Lecture:

Deadlock

- Necessary conditions of deadlock
- Deadlock prevention and avoidance
- Semaphore
- Mutex
- Producer consumer problem
- Dead-lock vs Starvation

No Lab

Software Development Methodologies

Duration: 46 hours (32 theory hours + 14 lab hours)

Objective: To build knowledge of Software development methodologies.

Evaluation: 65 marks (CCEE: 25 + Lab exam: 20 + Internals: 20)

Reference Books:

- Software Engineering by Chandramouli / Pearson
- Software engineering by Ian Sommerville / Pearson
- Object-Oriented Analysis and Design Using UML - An Introduction to Unified Process and Design Patterns by Mahesh P. Matha / PHI
- Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin / Prentice Hall
- The Mythical Man-Month: Essays on Software Engineering by Frederick P. Brooks Jr. / Addison Wesley
- User Stories Applied: For Agile Software Development by Mike Cohn / Addison Wesley
- DevOps: Continuous Delivery, Integration, and Deployment with DevOps by Sricharan Vadapalli / Packt
- Git for Teams by Emma Westby / O'Reilly

(Note: Each Session is of 2 hours)

Git (4 hours)

Session 1

Lecture

- Developing an application in a team
- Issues developers face when working in a team
- Introduction to code versioning system
- History of code versioning system

    - Different tools available for versioning
    - Software development workflow
- Introduction to git
- Introduction to git repository and git structure
- Adding code to git
- Creating and merging different git branches

Lab

- Create a local git repository
- Commit the initial code
- Update the code
- Use git commands to
    - Get the updated files
    - List the changes
    - Create branch
    - Merge branch

Software Engineering (10 hours)

Sessions 2, 3 & 4
Lecture

- Introduction to software engineering
    - Software Process
    - Software Process Model
    - Software Product
- Importance of Software engineering
- Software Development Life Cycles
- Requirements Engineering
    - Types of Requirements
    - Steps involved in Requirements Engineering
    - Requirement Analysis Modelling
- Design and Architectural Engineering
    - Characteristics of Good Design
    - Function Oriented vs Object Oriented System
    - Modularity, Cohesion, Coupling, Layering
    - Design Models
    - UML
- Coding
    - Programming Principles
    - Coding Conventions
- Object Oriented Analysis and Design

No Lab

Sessions 5 & 6
Lecture

- Introduction to Agile development model
- Agile development components
- Benefits of Agile
- Introduction to different tools used for agile web development
- Scrum and Extreme Programming
- Introduction to Atlassian Jira
    - Add Project
    - Add Tasks and sub-tasks

    - Create sprints with tasks
- Case study of developing web application using agile methodology

No Lab

DevOps (16 hours)

Sessions 7 & 8

Lecture

- Introduction to Microservices
- Microservices Architecture
- Fragmentation of business requirement
- Deployment pattern
- API gateway
- Service Discovery
- Database Management for Microservices

No Lab

Sessions 9 & 10

Lecture

- Introduction to DevOps
- DevOps ecosystem
- DevOps phases
- Introduction to containerisation
- Introduction to docker
- Creating docker images using Dockerfile
- Container life cycle

Lab

- Install and configure docker
- Create docker image using Dockerfile
- Start docker container
- Connect to docker container
- Copy the website code to the container
- Use docker management commands to
    - List the images
    - List the containers
    - Start and stop container
    - Remove container and image

Session 11

Lecture

- Introduction to YAML
- Introduction to Docker Swarm and Docker Stack
- Introduction to Kubernetes
- Creating Kubernetes cluster
- Creating service in Kubernetes
- Deploying an application using dashboard

Lab

- Configure Kubernetes
- Configure Kubernetes Dashboard
- Setup a Kubernetes cluster
- Access application using Kubernetes service
- Deploy the website using Dashboard

Testing & Integration (16 hours)

Session 12
Lecture
- Introduction to software testing
- Why testing code is important
- Verification and validation
- Quality Assurance vs Quality Control vs Testing
- Principles of software testing

Assignment
- Read more testing concepts used in the industry

Session 13
Lecture
- Introduction to STLC and V Model
- Types of testing: manual and automation
- Tools used for automation testing
- Introduction to testing methods: white-box, black-box and grey-box
- Introduction to functional testing: (* students are supposed to learn the concepts)
- Introduction to non-functional testing: (* students are supposed to learn the concepts)

Assignment
- Create a test plan for project
- Document the use cases
- Create test case document for different sprints (designed in SE)

Sessions 14 & 15
Lecture
- Introduction to Selenium (use Eclipse IDE)
- Load web driver
- Create selense commands: locators: by ID, name, class, tag name, XPath
- Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select

Lab
- Download and configure Selenium
- Create a test suite
- Add commands and interactions

Session 16
Lecture
- Introduction to delivery pipeline
- Introduction to Jenkins
- Jenkins management
- Adding slave node to Jenkins
- Building a delivery pipeline
- Selenium integration with Jenkins

Lab
- Install and configure Jenkins
- Build a pipeline job using Jenkins
- Create a maven project for Selenium
- Add Selenium test suite in the project
- Integrate it with Jenkins

Teaching Guidelines for

Object Oriented Programming with Java

PG-DAC September2023

Duration: 112 hours (50 theory hours + 50 lab hours + 12 revision/practice hours)

Objective: To reinforce knowledge of Object Oriented Programming concepts using Core Java.

Prerequisites: Basic knowledge of computer programming

Evaluation: Total 100 marks
Weightage: CCEE – 40%, Lab exam – 40%, Internals – 20%

Text Book:

- Core and Advanced Java Black Book / Dreamtech Press

References:

- Java 8 Programming Black Book / Dreamtech Press
- Core Java : Volume 1 - Fundamentals by Cay S. Horstmann / Prentice Hall
- Core Java : Volume 2 - Advanced Features by Cay S. Horstmann / Prentice Hall
- Programming in Java by Sachin Malhotra, Saurabh Choudhary / Oxford University Press
- Java The Complete Reference by Herbert Schildt / McGraw Hill
- Core Java 8 for Beginners by Sharanam Shah, Vaishali Shah / Shroff Publishers
- Murach's Java Programming by Joel Murach / Mike Murach
- Object-Oriented Analysis and Design with applications by Grady Booch / Pearson

(Note: Each Session is of 2 hours)

Session 1: Introduction to Java
Lecture:

- Introduction to java
- Features of java
- JVM Architecture
- JDK and its usage
- Structure of java class
- Working with data types: Primitive data types

Session 2: Basic programming concepts
Lecture:

- Java Tokens
- Declaring variables and methods
- Data type compatibility
- Operators
- Control statements
- Arrays 1-D and multidimensional array

Lab 1 & 2:

- Get yourself acquainted with java environment.
- Print different patterns of asterisk (*) using loops (e.g. triangle of *).

Tutorial:

- Compare syntactical similarities and dissimilarities between Java and C++.

<u>Object Oriented Programming Concepts</u>

Session 3: Object Oriented Programming Concepts
Lecture:

- Introduction to OOP
- Classes and Objects
- OOP principles
- Encapsulation, Abstraction, Inheritance and Polymorphism

Session 4:
Lecture:

- Static variables and methods
- Accessing static variables and methods of different class
- Introduction to reference data types
- Reference variables and methods
- Difference between reference data types and primitive data types
- Difference between reference variable and static variable

Session 5:
Lecture:

- Constructors, initializing reference variables using constructors.
- Pass by value v/s pass by reference.
- Re-assigning a reference variable.
- Passing reference variable to method
- Initializing reference variable of different class
- Heap memory and stack memory

Lab 3 & 4:

- Print default values of static & instance variables for different data types.
- Build a class Employee which contains details about the employee and compile and run its instance.
- Build a class which has references to other classes. Instantiate these reference variables and invoke instance methods.

Tutorial:

- Understand role of stack and heap memory in method invocation and object creation.

Session 6:
Lecture:

- Inheritance: single & multilevel
- Inheritance: Hierarchical
- Association, Aggregation and Composition

- Polymorphism: Compile time and runtime polymorphism
- Rules of overriding and overloading of methods
- super and this keyword

Lab 5 & 6:

- Create a class Employee and encapsulate the data members.
- Create demo applications to illustrate different types of inheritance.

Session 7:

Lecture:

- Upcasting &down casting of a reference variable
- Abstract class and abstract methods
- Interface (implementing multiple interfaces)

Session 8:

Lecture:

- Final variables, final methods and final class
- Functional interface
- New interface features (Java 8 & 11)
- Lambda Expression
- Inner Class (Regular, Method local, Anonymous & static inner class)
- Enum

Lab 7 & 8:

- Create an Array of Employee class and initialize array elements with different employee objects.
- Try to understand the no of objects on heap memory when any array is created.

Session 9:

Lecture:

- Access modifiers (public, private, protected and default)
- Packages and import statements.
- Static imports
- Constructor chaining (with and without packages)
- Accessing protected variables and methods outside the package

Session 10:

Lecture:

- Garbage collection in java
- Requesting JVM to run garbage collection.
- Different ways to make object eligible for garbage collection: (Nulling a reference variable, Re-assigning a reference variable & island of isolation)
- Finalize method.

Lab 9 & 10:

- Create a demo application to understand the role of access modifiers.
- Implement multilevel inheritance using different packages.
- Access/invoke protected members/methods of a class outside the package.
- Override finalize method to understand the behavior of JVM garbage collector.

Sessions 11 & 12:
Wrapper Classes and String Class
Lecture:

- Wrapper classes and constant pools
- String class, StringBuffer& StringBuilder class
- String pool

Lab 11 & 12:

- Create sample classes to understand boxing & unboxing.
- Use different methods of java defined wrapper classes.
- Create StringDemo class and perform different string manipulation methods.

Tutorial:

- Understand the difference between String / StringBuffer / StringBuilder.

Sessions 13 & 14:
Exception Handling
Lecture:

- Exception hierarchy, Errors, Checked and un-checked exceptions.
- Exception propagation
- try-catch-finally block, throws clause and throw keyword.
- Multi catch block.
- Creating user defined checked and unchecked exceptions.

Lab 13 & 14:

- Create user defined checked and unchecked exceptions.

Session 15:
java.io & java.nio Package
Lecture:

- Brief introduction to InputStream, OutputStream, Reader and Writer interfaces
- NIO package
- Serialization and de-serialization
- Shallow copy and deep copy

Session 16:
Lecture:
Object Class & java.util Package

- Date, DateTime, Calendar class
- Converting Date to String and String to Date using SimpleDateFormat class
- Object Class: Overriding to String, equals &hashcodemethod

Lab 15 & 16:

- Create a Demo class to Read & write image/text files.
- Create SerializationDemo class to illustrate serialization and de-serialization process.
- Create a demo class for Date, Time and Calendar

<u>Collections</u>

Sessions 17, 18 & 19:
Lecture:

- Introduction to collections: Collection hierarchy
- List, Queue, Set and Map Collections
- List Collection:
  - ArrayList, LinkedList
  - Vector (insert, delete, search, sort, iterate, replace operations)
- Collections class
- Comparable and Comparator interfaces
- Queue collection

Labs 17, 18 & 19:

- Create DateManipulator class to convert String to date, date to String and to find out number of days between two dates.
- Create a list of java defined wrapper classes and perform insert/delete/search/iterate/sort operations.
- Create a collection of Employee class and sort objects using comparable and comparator interfaces.
- Implement Queue data structure using LinkedList and Queue collection.

Sessions 20 & 21:
Lecture:

- Set Collection:
  - HashSet, LinkedHashSet&TreeSet collection
  - Backed set collections.
- Map Collection:
  - HashTable, HashMap, LinkedHashMap&TreeMap classes
  - Backed Map collections.
- Concurrent collections

Labs 20 & 21:

- Create an Employee HashSet collection and override equals &hashCode methods to understand how the set maintains uniqueness using these methods.
- Create a Sample class to understand generic assignments using "? extends SomeClass" , "? super someclass " and "?".

Session 22:
Lecture:

- MultiThreading : Thread class and Runnable Interface
- sleep, join, yield, setPriority, getPrioritymethods.
- ThreadGroup class

Lab 22:

- Create multiple threads using Thread class and Runnable interfaces.
- Assign same task and different task to multiple threads.
- Understand sleep, join, yield methods.

Sessions 23 & 24:

Lecture:

- Synchronization
- Deadlock
- Wait, notify and notifyAllmethods.
- Producer & Consumer problem

Lab 23 & 24:

- Create a Deadlock class to demonstrate deadlock in multithreading environment.
- Implement wait, notify and notifyAll methods.
- Demonstrate how to share threadlocal data between multiple threads.

Session 25 : Generics and Reflection API

Lecture:

- Introduction to generics
- Generic classes
- Generic methods
- Wild cards (upper and lower)
- Reflection

Lab 25:

- Invoke private methods of some other class using reflection.
- Create multiple threads using anonymous inner classes.
- Create multiple threads using lambda expressions.

Teaching Guidelines for
Algorithms and Data Structures Using Java
PG-DAC September 2023

Duration: 72 hours (32 theory hours + 32 lab hours + 8 revision/practice hours)

Objective: To reinforce knowledge of problem solving techniques, data structure concepts and analysis of different algorithms using Java.

Prerequisites: Knowledge of programming in C/C++ with object oriented concepts

Evaluation: 100 Marks
Weightage: CCEE – 40%, Lab exam – 40%, Internals & Mini project – 20%

Text Book:
- Data Abstraction and Problem Solving with Java: Walls and Mirrors by Janet Prichard , Frank M. Carrano / Pearson

References:
- Problem Solving: Best Strategies to Decision Making, Critical Thinking and Positive Thinking by Thomas Richards / Kindle Edition
- Data Abstraction and Problem Solving with Java: Walls and Mirrors by Janet Prichard , Frank M. Carrano / Pearson
- Object-oriented Analysis and Design Using UML - An Introduction to Unified Process and Design Patterns by Mahesh P. Matha / PHI
- Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein

---

(Note: Each Session is of 2 hours)

Session 1:
Problem Solving & Computational Thinking
Lecture:
- Define the problem
- Identify the problem
- Introduction to Problem Solving
- Problem solving basics

Lab:
- Faculty needs to assign different problems, mostly real world problems
- Students (team-wise, two students in a team) need to analyze as per the techniques learned
- Based on the above problems students need to select as per the selection criteria learned
- They need to implement the selected solution and need to do the documentations.
- Introducing the mini project ideas

Sessions 2 & 3:
Algorithms & Data Structures
Objective: At the end of the session students should know, what is the importance of data structure in problem solving. How stacks, queues, circular queues work. Their real world applications. How to solve problems using these data structures.

Lecture:

- Introductory Concepts
- Algorithm Constructs
- Complexity analysis of algorithms (Big O notation)
- OO design: Abstract Data Types (ADTs)
- Basic Data Structures
    - Arrays
    - Stacks
    - Queues
    - Circular Queues

Lab:

- Implement stack through array
- Complexity analysis of loops and recursive algorithms.
- Implement queues with inserting element at different location (First, Last)
- Implement circular queue

Sessions 4 & 5:
Linked List Data Structures
Objective: At the end of the session students should know, what are applications of Linked List, different types of link list. Comparison with arrays as when to use linked list and when to use array.
Lecture:

- Linked lists
    - Singly linked lists
    - Doubly linked lists
    - Circular linked lists
    - Node-based storage with arrays

Lab:

- Implement circular queue using linked list
- Implement stack using linked list

Session 6:
Recursion
Objective: At the end of the session students should know what is recursion, type of recursion, local variable in recursion, stack manipulation during recursion, function complexity
Lecture & Lab:

- What is recursion?
- What is the base condition in recursion.
- Direct and indirect recursion.
- Memory is allocated to different function calls in recursion.
- Pro and cons of recursion
- Function complexity during recursion

Sessions 7 & 8:
Trees & Applications
Objective: At the end of the session students should know what is the use of binary trees, how to create binary search trees. Different tree traversals. What are the applications of binary trees? How to calculate search complexity in binary search trees? What are the limitations of binary search trees? What are different options to overcome the binary search tree limitations.

Lecture:

- Introduction to trees
- Trees and terminology
- Tree traversals
- Binary trees
- Complete binary trees / Almost complete binary tree (ACBT)
- Array implementation of ACBT
- Binary search trees
- AVL tree

Lab:

- Write a program to implement a binary search tree and the following operations on it:
    - Create()
    - Tree traversals - Breadth First Search, Depth First Search, Inorder(), Preorder(), Postorder()
    - Delete()

Sessions 9, 10 & 11:
Searching & Sorting Algorithms

Objective: At the end of the session students should know what are the different types of sorting and searching algorithms, why all the sorting algorithms are equally important despite different time/space complexity of the algorithms. How the complexity is calculated for each of them. How to pick a sorting algorithm given the nature of the data to be sorted.

Lecture:

- Objectives of Searching
    - The Sequential Search
    - Analysis of Sequential Search
    - The Binary Search
- Analysis of Binary Search
- Introduction to sorting
    - Selection sort
    - Insertion sort
    - Bubble sort
    - Heapsort
    - Mergesort
    - Quicksort
- Analysis of sorting algorithms

Lab:

- Writing program to search an item through sequential search technique.
- Implement to find an item in a list through binary search
- Implement sorting algorithm for: insertion sort, Quicksort

Session 12:
Hash Functions and Hash Tables

Objective: At the end of the session students should know what is hashing, what is the importance of hashing, comparative complexity of hashing with other search techniques. Problems (collision) with hashing and what are the different solutions of that.

Lecture:

- Hashing & Introduction to hash tables
- Hash Functions
- Different type of hash functions

- Collision Resolution
- Linear Probing
- Quadratic Probing
- Double Hashing
- Inserting and Deleting an element from a hash table

Lab:

- Implement hashing techniques in different programs solved earlier
- Write a program to implement Hash table
- Fibonacci recursive algorithm improvement using hash table

Sessions 13 & 14:
Graphs & Applications
Objective: At the end of the session students should know what is graph? Why is graph the most generic data structure? Different types of graphs. Different representation of graph? Graph traversals (Breadth First Traversal, Depth First Traversal). Different applications which can be solved with graphs, real world and programming problems with graphs.
Lecture:

- Introduction to graph theory
- Graph Terminology
- Different types of Graphs
- Representation of Graphs
    - o Adjacency Matrix
    - o Adjacency List
    - o Graph Traversal Algorithms (Breadth First Search, Depth First Search)
- Shortest Path
    - o Level Setting : Dijkstra's algorithm
    - o Level Correcting: All-pairs shortest path, Floyd-Warshall algorithm
- Spanning Trees
    - o Minimum spanning tree algorithms,
    - o Prim's algorithm
    - o Kruskal's Algorithm

Lab:

- Implement a graph using adjacency Matrix and traverse using Depth First Search.
- Implement a graph and do traversal using stack and queue.

Sessions 15 & 16:
Algorithm Designs
Objective: At the end of the session students should know what are different classes of algorithms. What is the nature of each class of algorithms? How to pick an algorithm for a particular problem. What problems fall under each class of algorithms. What are the worst case, average case and the best case for algorithms?
Lecture:

- What are the different class of algorithms
- How to write efficient Algorithm
- Introduction to algorithm design techniques
- Algorithm Design techniques
- Analysis of an Algorithm
    - o Asymptotic Analysis
    - o Algorithm Analysis

- Analysis of different type of Algorithms
    - Divide and Conquer Algorithm
    - Greedy algorithm
    - Dynamic Programming algorithm
    - Brute force algorithm
    - Backtracking algorithms
    - Branch-and-bound algorithms
    - Stochastic algorithms
- Complexity
    - Complexity Analysis
    - Space complexity of algorithm
    - Time complexity of algorithm
- Case study on Algorithm Design techniques
- Application of Data structures

Lab + Assignment:

- Study on different Algorithms
- Compare different Algorithms previously programmed and do the analysis

Teaching Guidelines for
Database Technologies
PG-DAC September 2023

Duration: 72 hours (32 theory hours + 32 lab hours + 8 revision/practice hours)

Objective: To introduce students to RDBMS and NoSQL Databases and facilitate hands-on experience on SQL (using MySQL) and MongoDB.

Prerequisites: Working knowledge of Windows and Linux, familiarity with programming.

Evaluation: 100 Marks
Weightage: CCEE – 40%, Lab exam – 40%, internals – 20%

Text Book:
- Murach's MySQL by Joel Murach / Shroff Publisher

References:
- Database System Concepts by Abraham Silberschatz, Henry Korth and S. Sudarshan / McGraw Hill
- Database Design and Relational Theory: Normal Forms and All That Jazz by C. J. Date (Author) / O'Reilly
- Fundamentals of Database System by Shamkant B. Navathe, Ramez Elmasri / Pearson
- MySQL: The Complete Reference by Vikram Vaswani / McGraw Hill
- SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management by Andreas Meier and Michael Kaufmann / Springer
- MongoDB: The Definitive Guide by Shannon Bradshaw, Eoin Brazil and Kristina Chodorow / O'Reilly
- http://bigdata.stratebi.com/?language=en

(Note: Each Lecture and Lab Session is of 2 hours)

Session 1:
Lecture
Introduction to DBMS, Basic Database Terminology
Types of DBMS: Relational, Object Relational and NoSQL Databases
Introduction to MySQL, MySQL Clients (Monitor, Shell, Workbench)
Lab
Using MySQL Monitor, Shell, and Workbench

Session 2:
Lecture
Data Models (Conceptual, Logical, Physical)
Database Design, Entity-Relationship Diagram (ERD)
Codd’s 12 rules for RDBMS
Introduction to SQL, Categories of SQL Commands: DDL, DML, DCL, DTL/TCL
DDL (CREATE/ALTER/DROP/TRUNCATE)
Lab
Performing basic CREATE, ALTER, DROP Commands

Session 3:
Lecture
Data Redundancy, Data Anomalies, Functional Dependency
Normalization, Need for Normalization
Normal Forms (1st NF, 2nd NF, 3rd NF, BCNF) with examples, Introduction to 4th and 5th NF
DML (INSERT/UPDATE/DELETE)
Lab
DML (INSERT/UPDATE/DELETE), TRUNCATE

Session 4:
Lecture
MySQL Data Types, Database Constraints (Primary Key, Unique, Not Null, Foreign Key, Default, Check*)
Aggregate Functions, Grouping Things Together (Group By, Having)
LIKE Operator, DISTINCT, Sorting (Order by clause)
BETWEEN… AND Operators, Comparing Nulls (IS NULL/IS Not NULL), IN/NOT IN
Lab
Defining Data Types for Columns
Creating, Altering, Dropping Constraints
Aggregate Functions: SUM(), AVG(), COUNT(), MAX(), MIN(), COUNT(), Group By, Having Clause
Using Like, Distinct, Order By, Between...And
Comparing Nulls, Using IN/Not-In

Session 5:
Lecture
Relational Algebra Operations (Selection, Projection, Union, Intersect*, Minus*, Cross/Cartesian)
Joins (Eqvi, Inner, Outer, Natural, Cross), SQL Standard Syntax for Joins
Copying table structure/data, Sequences (AUTO_INCREMENT)
Lab
Union/Union ALL
Queries on Various type of Joins using OLD and SQL Standard Syntax
Copying table structure, Copying data from one table to another
Using AUTO_INCREMENT

Session 6:
Lecture
Subquery, Correlated Subquery, EXISTS/NOT EXISTS
TCL Commands (Commit/Rollback/Savepoint), DCL Commands (GRANT/REVOKE/GRANT OPTION)
Views, Types of Views, Simple and Complex Views
Lab
Subqueries, Correlated Queries
Using Exists/Not-Exists
Using Commit/Rollback/Savepoint
Granting/revoking privileges on database objects
Creating Views, Querying using Views
Creating Indexes
Creating Temporary Tables

Session 7:
Lecture
Indexes, Benefit of Indexes, Type of Indexes, Temporary Tables
ACID Properties, Concept of Database Instance and Schema
MySQL Storage Engines (InnoDB, MyISAM and others),
Lab
Indexes, Temporary Tables
All other SQL Commands Revision

Session 8:
Lecture
Introduction to MySQL Programming, Use of MySQL Programs,
Introduction to Stored Procedures, Benefits of Stored Procedures
Procedure Parameters (IN, OUT and INOUT).
Lab
Creating procedure without parameters
Creating Procedure with (IN/OUT/INOUT) Parameters

Session 9:
Lecture
Flow Control Statements (LOOP, WHILE and REPEAT)
Using above statements in Stored Procedures/Functions
Conditional Statements (IF, IF-ELSE-THEN, SWITCH CASE)
Example of each type of statement
Lab
Use of flow control statement in Stored Procedure
Use of conditional statements in Stored Procedure

Session 10:
Lecture
Loop constructs (ITERATE, LEAVE)
Functions with and without parameters
MySQL Built-in functions (string, numeric, date etc.)
Lab
Creating Function and returning value from it
Use of built-in functions in queries

Session 11:
Lecture
Cursors (Asensitive, Insensitive, Read only, Nonscrollable)
Cursors example and real time use
Lab:
Writing procedures with Declare, fetch and close cursor
Example of each type of cursors

Session 12:
Lecture
Triggers (BEFORE, AFTER), New and Old trigger variables
Trigger Examples and real time use
Lab
CreateBefore Triggers
Create After Triggers

Session 13:
Lecture
Error Handling and Exceptions, Types of Handler Actions, How to write Handler
Defining and handling exceptions in Stored Procedures and Functions
Lab
Exception handling in Stored Procedure
Exception handling with various handler actions

Session 14:
Lecture
Introduction to NoSQL database, Features of NoSQL Database
Structured vs. Semi-structured and Unstructured Data
Difference between RDBMS and NoSQL databases
CAP Theorem, BASE Model
Categories of NoSQL Databases: Key-Value Store, Document Store, Column-Oriented, Graph
Introduction to MongoDB, Features of MongoDB
MongoDB command interface and MongoDB compass
Lab
Using MongoDB Shell and Compass

Session 15 & 16:
Lecture
MongoDB Documents & Collections
RDBMS & MongoDB analogies: relations/tables => collections; tuples/records => documents
JSON and BSON documents
Performing CRUD (CREATE, READ, UPDATE, DELETE) Operations, UPSERT
MongoDB – Operators, Sorting, Indexing
Lab:
Creating database, Connecting to a database, Creating Collections
Performing CRUD operations
MongoDB: Complex Read Using Operators, Sorting Operations, Creating Indexes