3rd Semester

S. No.	Course Code	Course Title	Credit	Contact Hours			
				L	T	P	Total
1	ITT201	Data Structures	4	3	1	0	4
2	ITT202	Signal and Systems	3	3	1	0	4
3	ITT203	Software Engineering	3	2	1	0	3
4	ITT204	Discrete Mathematics and Graph Theory	4	3	1	0	4
5	CST201	Object Oriented Programming I	4	3	1	0	4
6	ECT207	Electronics	4	3	1	0	4
7	CSL203	Object Oriented Programming Lab I	1	0	0	2	2
8	ECL208	Electronics Lab	1	0	0	2	2
9	ITL205	Data Structures Lab	1	0	0	2	2
Total			**25**	**17**	**6**	**6**	**29**

4th Semester

S. No.	Course Code	Course Title	Credit	Contact Hours			
				L	T	P	Total
1	ITT250	Operating Systems	4	3	1	0	4
2	ITT251	Database Management System	4	3	1	0	4
3	ECT251	Digital Electronics & Logic Design	4	3	1	0	4
4	ECT253	Communication System	4	3	1	0	4
5	EET258	Control System	3	3	0	0	3
6	ECL254	Digital Electronics & Logic Design Lab	1	0	0	2	2
7	ECL255	Communication System Lab	1	0	0	2	2
8	ITL252	Operating Systems Lab	1	0	0	2	2
9	ITL253	Database Management System Lab	1	0	0	2	2
10	ITL254	Web Programming	2	0	0	4	4
Total			**25**	**15**	**4**	**12**	**31**

<table>
<tr><td rowspan="3"><b>Subject: Data Structures (Code: ITT201)</b></td><td colspan="2" rowspan="3"><b>Year &amp; Semester: B. Tech Information Technology Engineering 2nd Year & 3rd Semester</b></td><td colspan="3"><b>Total Course Credit: 4</b></td></tr>
<tr><td>L</td><td>T</td><td>P</td></tr>
<tr><td>3</td><td>1</td><td>0</td></tr>
<tr><td rowspan="2"><b>Evaluation Policy</b></td><td>Mid-Term</td><td>Class Assessment</td><td colspan="3">End-Term</td></tr>
<tr><td>30 Marks</td><td>10 Marks</td><td colspan="3">60 Marks</td></tr>
</table>

Course Outcomes (COs):

CO1: Understand basic data structures such as arrays, strings, and linked lists.

CO2: Study linear data structures such as stacks and queues and understand their difference.

CO3: Describe the hash function and concepts of collision and its resolution methods.

CO4: Understand the concept of memory management.

CO5: Study tree, heap and graphs along with their basic operations.

CO6: Study different techniques for solving problems like sorting and searching

Syllabus:

Introduction: Basic concept of data, structures and pointers.

Arrays: Representation, implementation, polynomial representation. Limitations.

Strings: Representation, String operations, Implementing String.h library functions.

Linked List: Static and dynamic implementation. Single, double, circular, multiple linked lists.

Stacks: Recursion and Stacks. Static and dynamic implementation. Expression evaluation. Infix, postfix expressions, multiple stacks.

Queues: Static and dynamic implementation, circular queues, and implementation.

Hash Tables: Hash tables implementation. Hashing techniques, single, double.

Storage Management: Memory Management techniques, garbage collection.

Trees: Binary trees, binary search trees, static and dynamic implementation. Tree operations, insert, delete, and search.

Heaps: Implementation, sorting etc.

Sorting and Searching: Different sorting techniques. Insertion sort, selection sort, bubble sort, radix sort, quick sort, merge sort, heap sort.

Graphs: Representation of graphs, BFS, DFS sort. Graph Algorithms.

Books Recommended:

1. Data Structures by Rajni Jindal

2. Data Structures - Schaum‘s Series

3. Data Structures by Knuth

4. Data Structures by Farouzan

5. Data Structures using C and C++ by Langsam, Augestern, Tanenbaum.

Subject: Signals & Systems (Code: ITT202)	**Year & Semester: B. Tech Information Technology Engineering 2nd Year & 3rd Semester**		**Total Course Credit: 3**		
			L	T	P
			2	1	0
Evaluation Policy	Mid-Term	Class Assessment	End-Term		
	30 Marks	10 Marks	60 Marks		

<u>Course Outcomes (COs):</u>

CO1. Understand the basic classification, properties and operations on signals

CO2. Understand the basic classification, properties of systems and LTI systems.

CO3. Finding the Fourier and inverse transform of signals and its properties

CO4. Finding the Laplace and Inverse transform of signals and its properties.

CO5. To understand use of probability distribution function and density for signals and Systems

CO6. Using autocorrelation and cross correlation functions and noise in LTI systems

Syllabus:

1. Introduction to signals:

Classification of signals; Deterministic and non-deterministic, periodic and aperiodic, even and odd signals, energy and power signals, elementary signals; exponential, sinusoidal, impulse, step, ramp, pulse, square wave signals. Time shifting, time scaling and timeinversions of signals

2. Linear Time invariant systems

Continuous time system, basic system properties like causality, time invariance, stability, linearity, memory, order of system, interconnection of systems, Linear time invariant systems, characterization, unit impulse response , convolution, properties of LTI systems, linear constant co-efficient differential equations and system description.

3. Fourier analysis of signals and systems

Fourier series of periodic signals and its properties, Fourier transform of aperiodic signals and its properties, fourier transform of periodic signals, convolution in time and frequency domain, energy and signals, parsevals theorem, energy spectral density and its properties, Transfer function of LTI system.

4. The Laplace Transform

Definition, relation between Laplace and Fourier transforms, region of convergence, properties of Laplace transform, initial and final value theorems, convolution, transfer function of LTI system, concept of poles and zeroes, stability criteria.

5. Random variable theory and random signals

Probability, conditional probability, statistical independence, random variables, discrete and continuous random variables, probability distribution and probability density functions, statistical averages of random variables. Some important density functions.

6. Random processes and characterization

Ensemble and time averages, stationary and non-stationary random process, wide sense stationary random process, autocorrelation and cross-correlation functions, response of LTI systems to random inputs, noise and its types, white noise, signal to noise ratio of LTI systems.

Text Books:

1. Signals and Systems by Zieman, Tranter, Fannin

2. Signals and Systems by Sanjay Sharma

Reference Books:

1. Signals and Systems by A Populis
2. Random processes and Systems by A Populis
3. Signals and Systems by S. Hykin

<table>
<tr><td rowspan="3">**Subject: Software Engineering
(Code: ITT203)**</td><td colspan="2" rowspan="3">**Year & Semester: B. Tech Information Technology Engineering 2nd Year & 3rd Semester**</td><td colspan="3">**Total Course Credit: 3**</td></tr>
<tr><td>L</td><td>T</td><td>P</td></tr>
<tr><td>3</td><td>0</td><td>0</td></tr>
<tr><td rowspan="2">**Evaluation Policy**</td><td>Mid-Term</td><td>Class Assessment</td><td colspan="3">End-Term</td></tr>
<tr><td>30 Marks</td><td>10 Marks</td><td colspan="3">60 Marks</td></tr>
</table>

Course Outcomes (COs):

CO1 Students will know classical and evolving software engineering methods, can select and tailor appropriate methods for projects, and can apply them as both team members and managers to achieve project goals.

CO2 Ability to effectively apply software engineering practice over the entire system lifecycle. This includes requirements engineering, analysis, prototyping, design, implementation, testing, maintenance activities and management of risks involved in software and embedded systems.

CO3 To make students proficient in effective written and oral communication skills so that they can prepare and publish the necessary documents required throughout the project lifecycle. It also includes effectively contributing to project discussions, presentations, and reviews.

CO4 To make students knowledgeable of the ethics, professionalism, and cultural diversity in the work environment and develop an awareness of the role and responsibilities of the professional software Engineers and Understanding software testing approaches such as unit testing and integration testing along with the need for lifelong learning and readily adapting to new software engineering environments.

Syllabus:

Unit I - Software Process:

The Evolving role of Software , Defining Software, Software Myths, Legacy software, A generic view of process, A layered Technology, Process Framework, Capability Maturity Model Integration (CMMI), Process Assessment, Personal and Team Process Models, Product and Process, Process Models – Build and fix model, The Waterfall Model, Incremental Process Model, RAD Model, Evolutionary Process Models, Unified Process, Agile Methodology, SCRUM Approach.

Unit II - Requirement Engineering:

Software Engineering Practice, Requirements Engineering tasks, Types of requirements, Feasibility studies, initiating the requirements Engineering Process, Eliciting Requirements, Developing Use cases, Requirement Analysis, Documentation and validation, Building the Analysis Model, Elements of the Analysis Model **Unit III - Analysis Modeling And Project Planning:**

Requirements Analysis, Analysis Modeling approaches, data modeling concepts, Object oriented Analysis, Scenario based modeling, Flow oriented Modeling, Class based modeling, creating a behavior model. Planning: Size estimation, Cost estimation, COCOMO, Software risk management.

Unit IV - Design & Implementation:

Design Engineering, Design Concepts, Modularity, Strategy of Design, Function oriented Design, Architectural Design, Detailed Design, Design process, Design Quality, Design model, User interface Design, Implementation, issues in implementation. Software metrics, SCM.

Unit V - Testing & Maintenance:

Testing strategies, Testing Tactics and terminologies, functional testing, structural testing, levels of testing, validation testing, system testing, Art of debugging. Software maintenance, maintenance models, Regression testing, Reverse Engineering, ReEngineering, evolution, Quality Management, Process Improvement, Risk Management.

Text Books:

1. Ian Sommerville, “Software Engineering”, Pearson Education.
2. Software Engineering – A Practitioner’s Approach, Roger S. Pressman, McGraw Hill.
3. Software Engineering, K.K. Aggarwal, Yogesh Singh, New Age International Publishers.

Reference Books:

1. Richard Fairley, "Software Engineering Concepts", McGraw Hill.
2. Stephan Schach, “Software Engineering”, Tata McGraw Hill.
3. Pfleeger and Lawrence,“Software Engineering: Theory and Practice”, Pearson Education.

<table>
<tr><td rowspan="3"><b>Subject: Discrete Mathematics & Graph Theory
(Code: ITT204)</b></td><td colspan="2" rowspan="3"><b>Year &amp; Semester: B. Tech Information Technology Engineering 2nd Year & 3rd Semester</b></td><td colspan="3"><b>Total Course Credit: 4</b></td></tr>
<tr><td>L</td><td>T</td><td>P</td></tr>
<tr><td>3</td><td>1</td><td>0</td></tr>
<tr><td rowspan="2"><b>Evaluation Policy</b></td><td>Mid-Term</td><td>Class Assessment</td><td colspan="3">End-Term</td></tr>
<tr><td>30 Marks</td><td>10 Marks</td><td colspan="3">60 Marks</td></tr>
</table>

Course Outcomes (COs):

CO1 Understand discrete structures such as sets, relations, and lattices.

CO2 Study the basic operations of Propositional logic and Boolean Algebra.

CO3 Analyse and study various proof techniques.

CO4 Understand basics of Graph theory and how it can be used to visualize and simplify problems.

CO5 Understand the various properties of algebraic systems like Rings, Monoids and Groups.

Syllabus:

Unit I - Sets and Relations:

Sets, Relations, Representation of Relations, Composition of Relations, Partitions, Equivalence Relations. Ordered sets and Lattices: Ordered sets, Diagram of Partially ordered sets, Supremum and Infimum, well ordered sets, Lattices, Bounded and complemented lattice, Distributive Lattice.

Unit II - Propositional Calculus:

Statements, Basic operations, Truth value of compound statements, Algebra of Propositions, Tautologies and contradiction, Conditional and Bi-conditional statements, logical implications,logical equivalence, predicates, Universal and existential quantifiers.Logic gates, Boolean Algebra, Postulates of Boolean Algebra; Theorems of Boolean Algebra, Sum of products and product of sums Simplification.

Unit III - Proof techniques:

Notions of implication, converse, inverse, contrapositive, negation, and contradiction; the structure of formal proofs; direct proofs; proof by counterexample; proof by contraposition; proof by contradiction; mathematical induction.

Unit IV - Graph Theory:

Graphs and Multi-graphs, Degree of a vertex, Paths connectivity, Cut points Bridges, Walks, paths, cycles, connected graphs, Bipartite, Regular,Planar and connected graphs,Euler graphs, Euler's theorem, Hamiltonian path and circuits, Graph coloring, chromatic number, isomorphism and Homomorphism of graphs,Konigsberg seven bridge problem, Shortest path. Trees, properties of trees, pendant vertices in trees, Degree sequences in trees, Necessary and sufficient conditions for a sequence to be a degree sequence of a tree.

Unit V - Group Theory:

Groups, semigroup, infinite group, Finite group, order of a group, Abelian group, subgroup, Lagrange's Theorem, Cosets, Normal Subgroups, order of an element of a group, cyclic group. Rings , Homomorphism and Isomerphism of rings.

Books Recommended

1. C. L. Liu : Elements of Discrete Mathematics, 2nd Ed.Tata Mc-Graw Hill.
2. Kolman, Busby and Ross : Discrete Mathematical Structers, 6th Ed.PHI (2009).
3. NarsinghDeo : Graph Theory with Applications to Engineering and Computer Sciences, PHI.
4. Murry R. Spiegel: Discrete Mathematics (Schaums Outline series) Tata McGraw Hill (2009).
5. K. Bogart, S. Drysdale, C. Stein. Discrete Math for Computer Science Students.

Reference Books

1. Kenneth H. Rosen:Discrete Mathematics and its applications,Tata McGraw Hill (2003).
2. K.R Parthasarty : basic Graph Theory, Tata Mc-Graw Hill

Subject: Object Oriented Programming **(Code: CST201)**	**Year & Semester: B. Tech** **Information Technology Engineering** **2nd Year & 3rd Semester**	**Total Course Credit: 4**		
		L	T	P
		3	1	0
Evaluation Policy	Mid-Term/Class Assessment (40 Marks)	Final-Term (60 Marks)		

Course Objectives

·To explore the principles of Object Oriented Programming (OOP).

·To understand object-oriented concepts such as data abstraction, encapsulation, inheritance, dynamic binding, and polymorphism.

·To use the object-oriented paradigm in program design.

·To lay a foundation for advanced programming.

·Provide programming insight using OOP constructs

Learning Outcomes

On completion of the course, student will be able to:

·Analyze the strengths of object oriented programming

·Design and apply OOP principles for effective programming

·Develop programming application using object oriented programming language C++ ·Percept the utility and applicability of OOP.

Course Outline / Content	
Unit	**Topics**
1.	**Classes and Objects:** Need of Object-Oriented Programming (OOP), Object Oriented Programming Paradigm, Basic Concepts of Object-Oriented Programming, Benefits of OOP, C++ as object oriented programming language. C++ Programming- C++ programming Basics, Data Types, Structures, Enumerations, control structures, Arrays and Strings, Class, Object, class and data abstraction, class scope and accessing class members, separating interface from implementation, controlling access to members. Functions- Function, function prototype, accessing function and utility function, Constructors and destructors, Copy Constructor, Objects and Memory requirements, Static Class members, data abstraction and information hiding, inline function.

2.	**Polymorphism and Inheritance:** Operator Overloading- concept of overloading, operator overloading, Overloading Unary Operators, Overloading Binary Operators, Data Conversion, Type casting (implicit and explicit), Pitfalls of Operator Overloading and Conversion, Keywords explicit and mutable. Inheritance- Base Class and derived Class, protected members, relationship between base Class and derived Class, Constructor and destructor in Derived Class, Overriding Member Functions, Class Hierarchies, Inheritance, Public and Private Inheritance, Levels of Inheritance, Multiple Inheritance, Ambiguity in Multiple Inheritance, Aggregation, Classes Within Classes. Polymorphism- concept, relationship among objects in inheritance hierarchy, abstract classes, polymorphism.
3.	**Virtual Functions:** Virtual Functions- Pointers- indirection Operators, Memory Management: new and delete, Pointers to Objects, A Linked List Example, accessing Arrays using pointers, Function pointers, Pointers to Pointers, A Parsing Example, Debugging Pointers, Dynamic Pointers, smart pointers, shared pointers, Case Study : Design of Horse Race Simulation.
	Virtual Function- Friend Functions, Static Functions, Assignment and Copy Initialization, this Pointer, virtual function, dynamic binding, Virtual destructor.
4.	**Templates and Exception handling:** Templates- function templates, Function overloading, overloading Function templates, class templates, class template and Non-type parameters, template and inheritance, template and friends Generic Functions, Applying Generic Function, Generic Classes, The typename and export keywords, The Power of Templates. Exception Handling- Fundamentals, other error handling techniques, simple exception handlingDivide by Zero, rethrowing an exception, exception specifications, processing unexpected exceptions, stack unwinding, constructor, destructor and exception handling, exception and inheritance.

5.	**Files and Streams:** Data hierarchy, Stream and files, Stream Classes, Stream Errors, Disk File I/O with Streams, File Pointers, and Error Handling in File I/O, File I/O with Member Functions, Overloading the Extraction and Insertion Operators, memory as a Stream Object, Command-Line Arguments, Printer output, Early vs. Late Binding.
1.	**Standard Template Library:** Standard Template Library, Overview of Standard Template Library, Containers, Algorithms, Iterators, Other STL Elements, Container Classes, General Theory of Operation, Vectors.

Books Recommended

Text Books	1.Robert Lafore, “Object Oriented Programming in Turbo C++”, Galgotia Publications, 2.Balagurusamy, “Object Oriented programming with C++”, Tata McGraw Hill.
References	1.BjarneStrustrup, “The C++ programming Language”, Addison Wesley, 2.Booch, “Object Oriented Analysis and Design with Applications, Addison Wesley. 3.Chair H. Pappas & William H. Murray, “The Complete Reference Visual C++”, TMH.

Subject: Electronics (Code: ECT207)	Year & Semester: B. Tech Information Technology Engineering 2nd Year & 3rd Semester	Total Course Credit: 4		
		L	T	P
		3	1	0
Evaluation Policy	Mid-Term/Class Assessment (40 Marks)	Final-Term (60 Marks)		

Course Objectives

·To get basic idea about types, specification and common values of passive components.

·To familiarize the working and characteristics of diodes, transistors, MOSFETS and some measuring instruments.

·To understand working of diodes in circuits and in rectifiers.

Learning Outcomes

Student can identify the active and passive electronic components. Student can setup simple circuits using diodes and transistors. Student will get fundamental idea about basic communication systems and entertainment electronics.

Course Outline / Content

Unit	Topics
1.	**Semiconductors:** Insulators, semiconductors and metals, Mobility and conductivity, Intrinsic and extrinsic semiconductors, Charge Densities in Semiconductors, Mass action Law, Current Components in Semiconductors, The Continuity Equation, Injected minority Charge Carrier, Hall effect.
2.	**PN Junction Diode:** Characteristic and analysis, Types of diodes – Zener diodes, Photodiodes, Light emitting diodes (LED‟s), Varactor diodes and tunnel diodes. Rectifiers and filter circuit: Half wave, full wave and Bridge rectifier circuits and their analysis, L, C and Pi filters, Basic regulator supply using zener diode, Clipping and clamping circuits.
3.	**Transistors:** Construction and characteristics of bipolar junction, transistors (BJT‟s)-Comm. Base, Comm. emitter, Comm. Collector configuration. Transistor at low frequencies – small signal low frequency transistor model (hparameters). Analysis of transistor amplifier circuit using h-parameters. Transistor biasing and bias stabilization: the operating point, stability factor, analysis of fixed base bias, collector to base bias, Emitter resistance bias circuit and self bias circuit. Bias compensation techniques.
4.	**Field Effect Transistor:** Construction and characteristics of JFET, JFET biasing circuit, JFET amplifier, MOSFET construction and characteristics.
5.	**Amplifiers And Oscillators:** Classification of amplifiers, concept of feedback, general characteristics of feedback amplifiers, Single stage RC coupled amplifier. Oscillators – Criterion for Oscillation, type of oscillators: Hartley oscillator, Colpitt Oscillator, RC Phase shift oscillator, Crystal oscillator.
6.	**Operational Amplifiers:** Introduction to Op-amp, Inverting and non-inverting configuration, Applications – adder, subtractor, integrator, differentiator.

Books Recommended

Text Books	1. Bhargava N. N., D C Kulshreshtha and S C Gupta, "Basic Electronics & Linear Circuits", Tata McGraw Hill, 2/e, 2013. 2. Electronics Devices and Circuit Theory by R. Boylestad, Pearson.
References	1. Bell, D. A., Electronic Devices and Circuits, Oxford University Press. 2. Boylested, R. L. and Nashelsky, L., Electronic Devices and Circuit Theory, Pearson Education. 3. Frenzel, L. E., Principles of Electronic Communication Systems, McGraw Hill.

Subject: Object Oriented Programming Lab (Code: CSL203)	**Year & Semester: B. Tech Information Technology Engineering 2nd Year & 3rd Semester**	**Total Course Credit: 1**		
		L	T	P
		0	0	2
Evaluation Policy	Mid-Term/Class Assessment (40 Marks)	Final-Term (60 Marks)		

Course Objectives

·The student should be able to explain the fundamental properties of the C++ language.

·The student should be able to combine the elements of the C++ language in developing structured programs.

·The student should be able to demonstrate the skills necessary to correctly compile, debug, and test programs in C++.

Learning Outcomes

At the end of the course student will be able to:

·Apply C++ features to program design and implementation

·Explain object-oriented concepts and describe how they are supported by C++

·Use C++ to demonstrate practical experience in developing object-oriented solutions

·Analyse a problem description and design and build object-oriented software using good coding practices and techniques

·Use common software patterns in object-oriented design and recognise their applicability to other software development contexts.

Course Outline / Content

Unit	Topics
1.	Function overloading and default arguments in C++
2.	Simple class design in C++, namespaces, object creation
3.	Class design in C++ using dynamic memory allocation
4.	Constructors and destructors
5.	Operator overloading and friend functions
6.	Overloading assignment operator and type conversions
7.	Inheritance, run time polymorphism and virtual functions
8.	Template design in C++
9.	Interface and abstract classes
10.	Exception handling
11.	File handling in C++

Books Recommended

Text Books	1.Robert Lafore, “Object Oriented Programming in Turbo C++”, Galgotia Publications, 2.Balagurusamy, „Object Oriented programming with C++”, Tata McGraw Hill.
References	1.BjarneStrustrup, “The C++ programming Language”, Addison Wesley, 2.Booch, “Object Oriented Analysis and Design with Applications, Addison Wesley. 3.Chair H. Pappas & William H. Murray, “The Complete Reference Visual C++”, TMH.

Subject: Electronics Lab (Code: ECL208)	Year & Semester: B. Tech Information Technology Engineering 2nd Year & 3rd Semester	Total Course Credit: 1		
		L	T	P
		0	0	2
Evaluation Policy	Mid-Term/Class Assessment (40 Marks)	Final-Term (60 Marks)		

Course Objectives

·To familiarize with the electronic components and basic electronic instruments.

·To enable the students to understand the behaviour of semi conductor devices based on experimentation.

Learning Outcomes

·To make familiar with PCB design and various processes involved.

·Ability to understand and analyse, linear and digital electronic circuits.

Course Outline / Content	
Unit	**Topics**
1.	Characteristics of Semi conductor diode and Zener diode

2.	Characteristics of a NPN Transistor under common emitter, common collector and common base configurations
3.	Characteristics of JFET (Draw the equivalent circuit)
4.	Characteristics of UJT and generation of saw tooth waveforms
5.	Design and Frequency response characteristics of a common emitter amplifier
6.	Design and testing of RC phase shift, LC oscillators
1.	Single phase half-wave and full wave rectifiers
8.	a) To assemble a half wave and a full wave rectifier and to study their performance. b) To suppress the ripple using RC filter.
9.	To assemble and observe the performance of clipping and clamping circuits.
10.	Design and realize Inverting and Non-inverting amplifier using 741 Op-amp.

Books Recommended

Text Books	1.Bhargava N. N., D C Kulshreshtha and S C Gupta, "Basic Electronics & Linear Circuits", Tata McGraw Hill, 2/e, 2013. 2Electronics Devices and Circuit Theory by R. Boylestad, Pearson.

References	1.Bell, D. A., Electronic Devices and Circuits, Oxford University Press. 2.Boylested, R. L. and Nashelsky, L., Electronic Devices and Circuit Theory, Pearson Education 3.Frenzel, L. E., Principles of Electronic Communication Systems, McGraw Hill.

Subject: Data Structures Lab **(Code: ITL205)**	**Year & Semester: B. Tech Information Technology Engineering** **2nd Year & 3rd Semester**	**Total Course Credit: 1**		
		L	T	P
		0	0	2
Evaluation Policy	Mid-Term/Class Assessment (40 Marks)	Final-Term (60 Marks)		

Course Outcomes (COs):

CO1 Implement and understand linked lists for polynomial manipulation.

CO2 Study stack data structure and use it for expression parsing (prefix, infix, postfix).

CO3 Implement circular queues for producer-consumer problem simulation.

CO4 Implement search trees and priority queues using heaps.

CO5 Implement various hashing techniques.

CO6 Study and implement various searching and sorting techniques.

C07 Implement various Graph Algorithms

Syllabus:

Basic concepts of data, linear lists, strings, arrays and orthogonal lists, representation of trees

& graphs, storage systems, Arrays, Recursion, Stacks, Queues, Linked lists, Binary trees,

General Trees, Tree Traversal, Symbol Table and Searching Techniques,

Sorting Techniques, **Graphs:**

1. Implement singly and doubly linked lists.
2. Represent a polynomial as a linked list and write functions for polynomial addition.
3. Implement stack and use it to convert infix to postfix expression
4. Implement array-based circular queue and use it to simulate a producer-consumer problem.

5. Implement an expression tree. Produce its pre-order, in-order, and post-order traversals.

6. Implement a binary search tree.

7. Implement priority queue using heaps

9. Implement hashing techniques

10. Implement various sorting techniques as taught in class.

11. Implement Dijkstra's algorithm using priority queues.

12. Implement Prim's and Kruskal's algorithms