# COURSE PLAN FOR *2017-2018 ODD SEMESTER*

***Semester:* III**

|  |  |
| --- | --- |
| *Course Title*: **Data Structures** | *Course Code*: **3CCI02** |
| *Total Contact Hours*: **52 Hrs** | *Duration of SEE*: **3 Hrs** |
| *SEE Marks*: **50** | *CIE Marks*: **50** |
| *Faculty Name: G Bhaskar* |  |

## Prerequisites:

Knowledge of fundamentals of C programming, problem-solving, design, and implementation skills and basic mathematics is essential.

## Course Overview:

* The primary objective is to introduce the student to advanced level of computer programming with an emphasis on design and implementation of abstract data structures.
* Choose the data structures that effectively model the information in a problem.
* Judge efficiency trade-offs among alternative data structure implementations or combinations.
* To write Program effectively with pointers, arrays, structures, and dynamically allocated memory and describe their internal representations.
* To demonstrate understanding of the abstract properties of various data structures such as stacks, queues, lists, and trees.
* Comparisons of different implementations of data structures and to recognize the advantages and disadvantages of the different implementations.

**Course Learning Objectives – CLOs**

*At the end of the course the student should be able to:*

1. Describe the properties of various data structures such as stacks, queues, lists, and trees.
2. Implement the data structures such as stacks, queues, lists, and trees using C language.
3. Compare different implementations of Data Structures and to recognize the advantages and disadvantages of the different implementations
4. Explain the applications of various Data Structures.
5. Choose an appropriate Data Structure for a given problem.

**Course Outcomes - COs**

*Upon successful completion of the course the student will be able to:*

CO1: **Apply** advanced C programming techniques like pointers, dynamic memory allocation, structures to developing solutions for particular problems;

**CO2:Design** and implement abstract data types such as linked list, stack, queue and tree by using C as the programming language using static or dynamic implementations techniques.

**CO3: Decide, Design** and **Develop** a solution for a given problem by **Selecting** an appropriate data structure for a given open end problem.

**CO4:** **Implement and submit** comprehensive and continuous assignment modules on societal problems as a team event (PO: 11(L) and PO: 12(L))

**CO5:** Do a **Survey** on the complex implementation challenges related to the data structures in real world scenario.

**Programme Outcomes(POs)**

**Engineering Graduates will be able to:**

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solution in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**Program Articulation Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course** | **POs** | | | | | | | | | | | | **PSOs** | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **1** | **2** | **3** |
| **Data structures using C** | 1 | - | 2 | - | 2 | 1 | 1 | - | 2 | 2 | - | - | 2 | 1 | 3 |

**MAPPING BETWEEN COs and POs**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CO/PO** | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | PSO1 | | PSO2 | PSO3 | |
|  | | | | | | | | | | | | |  | |  | | |  |
| **CO1** | **1** | **1** |  | |  |  |  |  |  |  |  |  |  |  | | **1** | | |  |
| **CO2** | **1** | **1** | **1** | | **1** | **1** |  | **1** |  | **1** |  |  |  | **2** | | **1** | | |  |
| **CO3** |  |  |  | | **2** | **2** |  | **1** |  | **1** |  |  |  | **2** | | **2** | | | 1 |
| **CO4** |  |  |  | |  |  |  |  |  | **2** |  | **1** | **1** | **1** | | **1** | | | 1 |
| **CO5** |  |  |  | | **1** |  |  |  |  |  | **1** |  | **1** | **1** | | **1** | | |  |
| **CO (overall)** | **1** | **1** | **2** | | **2** | **2** |  | **1** |  | **2** | **1** | **1** | **2** | **2** | | **1** | | | 1 |

**Degree of compliance 1: Slight (low) 2: Moderate (medium) 3:Substantial( High)**

## SCHEDULE PLAN FOR ACADEMIC YEAR 2017-18

\* Please note that this schedule may change as we progress through the course material

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Course Content** | **Teaching supplement** | **Student activity** | **Status** | **Deviation details** |
| **UNIT - 1** | | | | | | |
| 1. | 31-JULY-2017 | Introduction to Data Structures | Black Board |  |  |  |
| 2. | 01-AUG-2017 | Review of C programming Concepts | Black Board |  |  |  |
| 3. | 02-AUG-2017 | **Structures, unions and Dynamic memory allocation:** Defining the structures, Declaring structure variables, Accessing structure members, | Black Board |  |  |  |
| 4. | 05-AUG-2017 | Structure initialization, copying and comparing structure variables, | Black Board |  |  |  |
| 5. | 07-AUG-2017 | operations on individual members, | Black Board |  |  |  |
| 6. | 08-AUG-2017 | Array of structures, Programs. | Black Board |  |  |  |
| 7. | 09-AUG-2017 | unions, size of structures, bit fields, | Black Board |  |  |  |
| 8. | 12-AUG-2017 | Dynamic memory allocation functions. | Black Board |  |  |  |
| 9. | 14-AUG-2017 | **File management in C:** Defining and opening a file, closing a file, | Black Board |  |  |  |
| 10. | 16-AUG-2017 | input/output operations on files, | Black Board |  |  |  |
| 11. | 19-AUG-2017 | Programs on files. | Black Board |  |  |  |
| 12. | 21-AUG-2017 | Error handling during I/O operations, | Black Board |  |  |  |
| 13. | 22-AUG-2017 | random access to files, | Black Board |  |  |  |
| 14. | 23-AUG-2017 | Command line arguments. | Black Board |  |  |  |
| **UNIT - 2** | | | | | | |  |
| 15. | 26-AUG-2017 | **The Stack:** Definition and Examples, representing Stacks in C | Black Board |  |  |  |
| 16. | 28-AUG-2017 | Example: Infix, Postfix, and Prefix. | Black Board |  |  |  |
| 17. | 29-AUG-2017 | **Recursion:** Recursive Definition and Processes, Recursion in C, | Black Board |  |  |  |
| 18. | 30-AUG-2017 | Writing recursive programs: The Towers of Hanoi Problem, Efficiency of Recursion. | Black Board | Assignment-I |  |  |
| 19. | 04-SEPT-2017 | Programs on Stack | Black Board |  |  |  |
| 20. | 05-SEPT-2017 | Programs on Stack | Black Board |  |  |  |
| 21. | 06-SEPT-2017 | **Queues and Lists:** The Queue and Its Sequential Representation: C implementation of Queues, | Black Board |  |  |  |
| 22. |  | Insertion, Deletion and Display operations, | Black Board |  |  |  |
| 23. | 08-SEPT-2017 | Types of Queues (Linear, Circular, Priority and Double Ended Queues). | Black Board |  |  |  |
| 24. | 11-SEPT-2017 | Programs on Queues | Black Board |  |  |  |
| **UNIT – 3** | | | | | | | 12-SEPT-2017 |
| 25. | 13-SEPT-2017 | **Queues and Lists Continued**  Linked lists : Inserting and removing nodes from a list , | Black Board |  |  |  |
| 26. | 23-SEPT-2017 | Linked implementation of stacks, getnode and freenode operations, | Black Board |  |  |  |
| 27. | 25-SEPT-2017 | linked implementation of queues, | Black Board |  |  |  |
| 28. | 26-SEPT-2017 | Examples of list operation, | Black Board |  |  |  |
| 29. | 27-SEPT-2017 | List implementation of priority queues, | Black Board |  |  |  |
| 30. | 28-SEPT-2017 | Header nodes. | Black Board | **First Test** |  |  |
| 31. | 03-OCT-2017 | **Lists in C:** Allocating and freeing dynamic variables, | Black Board |  |  |  |
| 32. | 04-OCT-2017 | Programs on lists | Black Board |  |  |  |
| 33. | 07-OCT-2017 | Linked lists using dynamic variables, | Black Board |  |  |  |
| 34. | 09-OCT-2017 | Queues as lists in C, | Black Board |  |  |  |
| 35. | 10-OCT-2017 | Examples of list operations in C, | Black Board |  |  |  |
| 36. | 11-OCT-2017 | Non- integer and non homogeneous lists, | Black Board | Open Book Test |  |  |
| 37. | 14-OCT-2017 | Comparing dynamic and array implementations of lists, | Black Board |  |  |  |
| 38. | 16-OCT-2017 | Implementing header nodes. | Black Board |  |  |  |
| **UNIT - 4** | | | | | | |
| 39. | 17-OCT-2017 | **Other List Structures:** Circular lists, | Black Board |  |  |  |
| 40. | 21-OCT-2017 | Stack as a circular list, | Black Board |  |  |  |
| 41. | 14-OCT-2017 | Queue as a circular list, | Black Board |  |  |  |
| 42. | 16-OCT-2017 | Primitive operations on circular lists, | Black Board |  |  |  |
| 43. | 17-OCT-2017 | The Josephus problem, | Black Board |  |  |  |
| 44. | 21-OCT-2017 | Header nodes, | Black Board |  |  |  |
| 45. | 25-OCT-2017 | Programs on circular lists | Black Board |  |  |  |
| 46. | 30-OCT-2017 | Primitive operations on Doubly linked list. | Black Board |  |  |  |
| 47. | 31-OCT-2017 | **Trees:** Operations on Binary Trees, | Black Board |  |  |  |
| **UNIT - 5** | | | | | | |
| 48. | 07-Nov-2017 | Applications of Binary Trees, Binary Tree Representations: Node representation of Binary Trees, | PPT Slides |  |  |  |
| 49. | 08-Nov-2017 | Internal and External Nodes, Implicit array representation of Binary Trees, | PPT Slides |  |  |  |
| 50. | 14-Nov-2017 | Binary tree traversals: inorder, preorder, postorder, | PPT Slides |  |  |  |
| 51. | 15-Nov-2017 | Threaded Binary Trees. Trees and Their applications: | PPT Slides |  |  |  |
| 52. | 18-Nov-2017 | C Representations of Trees, Tree Traversals, | PPT Slides |  |  |  |
| 53. | 20-Nov-2017 | General Expressions as Trees, | PPT Slides |  |  |  |
| 54. | 21-Nov-2017 | Evaluating an Expression Tree, | PPT Slides |  |  |  |
| 55. | 22-Nov-2017 | Constructing a Tree. | PPT Slides |  |  |  |
| 56. | 25-Nov-2017 | Constructing a Tree. | PPT Slides |  |  |  |

**Tutorial class details:**

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No** | **Date** | **Topics to be Covered** |
| 1. | 03-AUG-2017 | Review of C programming |
| 2. | 10-AUG-2017 | Review of C programming |
| 3. | 17-AUG-2017 | Programs on Stacks |
| 4. | 24-AUG-2017 | Programs on Stacks |
| 5. | 31-AUG-2017 | Programs on Queues |
| 6. | 07-SEPT-2017 | Programs on Queues |
| 7. | 14-SEPT-2017 | Programs on Singly Linked Lists |
| 8. | 12-OCT-2017 | Programs on Singly Linked Lists |
| 9. | 19-OCT-2017 | Programs on Doubly Linked Lists |
| 10. | 26-NOV-2017 | Programs on Circular Linked Lists |
| 11. | 09-NOV-2017 | Programs on Trees |
| 12. | 16-NOV-2017 | Programs on Trees |

**TEXT BOOKS:**

|  |  |  |
| --- | --- | --- |
| 1 | E. Balaguruswamy | **Programming in ANSI C,** 4th Edition, Tata McGraw-Hill Publications |
| 2 | Yedidyah Langsam, MosheJ. Augenstein, AaronM. Tenenbaum | Data structures using C and C++,PHI/Pearson, 2nd Edition .(Chapter 2, 3.1, 3.2, 3.3(only the Towers of Hanoi Problem), 3.5. 4.1(excluding Queue as an ADT), 4.2, 4.3(except array implementation of list, Limitations of array implementation, comparing dynamic and array implementations of list), 4.5(except addition of long positive integers using circular and doubly linked list), 5.1, 5.2(except choosing Binary Tree Representation, Traversal using a Father field, Heterogeneous Binary Trees), and 5.5) |

**REFERENCE BOOK:**

|  |  |  |
| --- | --- | --- |
| 1 | Jean-Paul Tremblay,  Paul G. Sorenson | An Introduction To Data Structures With Applications, 2nd edition, McGraw-Hill International Editions |

**Assessment:**

|  |  |  |
| --- | --- | --- |
| **Course outcome** | **Method of assessment** | **Assessed during** |
| CO1 | Based on marks obtained. | Quiz , Assignment,  Tests, SEE |
| CO2 | Marks obtained in Lab exercises. |
| CO3 | Open Book test(Quiz 3) |
| CO4 | Based on implementation and the design report | Continuous and Comprehensive Assignment module |
| CO5 | Based on Survey report | Survey Report |

# **UNIT I: MODULE WISE PLAN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Course Code & Title****: 3*CCI02, Data Structures** | | | | | |
| *Chapter Title:* **Structures and Unions, Array of structures, Unions, bit fields. File management: Defining & opening a file** | | *Planned Hours:***10L+02T Hrs** | | | |
| ***Learning Objectives*** | | |  |
| At the end of the chapter the student should be able to: | | |  |
| 1. | **Explain** a mechanism of packing data of different types of variables under a single name.(L2) | | |  |
| 2. | **Draw** the logical structure of the members of the structure.(L2) | | |  |
| 3. | **Differentiate** between an array and structures.(L3) | | |  |
| 4. | **Implementation** of theunions and bit fields.(L4) | | |  |
| 5. | **Differentiate** between the unions and structures**.**(L2) | | |  |
| 6. | **Define** a file, what are the different operations performed with file and their syntaxes (L2). | | |  |
| 7. | **Design a** new structure for a given application problem.(L3) | | |  |
| 8. | **Explain** the concepts of bit fields and its advantages. (L3). | | |  |
| 9. | **Write** programs various operations performed with files.(L3) | | |  |
| 10. | **Write** program by using command line arguments.(L3) | | |  |
|  | | |  |

# **UNIT 2: MODULE WISE PLAN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Course Code & Title*: **3CCI02, Data Structures** | | | | |
| *Chapter Title:* ***The Stack,Recursion,Queues*** | | *Planned Hours:***11L+03T Hrs** | | |
| ***Learning Objectives*** | | |  |
| At the end of the chapter the student should be able to: | | |  |
| 1. | **Distinguish** **between** stacks and queues.(L2) | |  |
| 2. | **Describe** the structure of a Stack and its applications.(L3) | |  |
| 3. | **Distinguish** **between** recursive and non-recursive algorithms. (L3) | |  |
| 4. | **Explain** primitive operations that can performed on stack.(L2) | |  |
| 5. | **Implementation** of stacks using structures.(L1) | |  |
| 6. | **List** various types of queues.(L1) | |  |
| 7. | **Distinguish** **between** ordinary queue and priority queue.(L3) | |  |
| 8. | **Explain** primitive operations that can performed on queues.(L2) | |  |
| 9. | **Write** a recursive functions for various problems. (L4) | |  |
| 10. | **Distinguish** **between** dequeue and circular queue.(L3) | |  |

# **UNIT 3: MODULE WISE PLAN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Course Code & Title:* **3CCI02, Data Structures** | | | | |
| *Chapter Title:* **Linked lists,Lists in C: Linear Data Structures and their linked Storage Representation** | | *Planned Hours:* **12L+03T Hrs** | | |
| ***Learning Objectives*** | | |  |
| At the end of the chapter the student should be able to: | | |  |
| 1. | **Explain** the need of the Dynamic memory allocation.(L3) | |  |
| 2. | **Explain** the different types of lists and their storage representation.(L1) | |  |
| 3. | **Implementation** of thestacks and queues using dynamic memory allocation.(L2) | |  |
| 4. | **Distinguish** between the array implementation and linked list representations.(L3) | |  |
| 5. | **Write**  the list examples using c to store both integer and non-integer form.(L4) | |  |
| 6. | **Define** a list and the basic operations performed on it.(L1) | |  |
| 7. | **Differentiate** between the static memory allocation and dynamic memory allocation.(L2) | |  |
| 8. | **Differentiate** between the calloc()and malloc dynamic memory allocation funcitons. (L2) | |  |
| 9. | **Write** programs to implement the singly linked list with the header node.(L4) | |  |
| 10. | **Write** programs to implement primitive operations with the singly linked list.(L4) | |  |

# **UNIT 4: MODULE WISE PLAN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Course Code & Title: : 3***CCI02, Data Structures** | | | | |
| *Chapter Title:* **Other List Structures: Linear Data Structures – Doubly and Circular linked lists and Applications** | | *Planned Hours:* **09L+02T Hrs** | | |
| ***Learning Objectives*** | | |  |
| At the end of the chapter the student should be able to: | | |  |
| 1. | **Define** doubly linked list.(L1) | |  |
| 2. | **Write** programs to implement the primitive operations with doubly linked list.(L6) | |  |
| 3. | **Differentiate** between circular linked lists and doubly linked list.(L2) | |  |
| 4. | **List** different types of lists.(L1) | |  |
| 5. | **Explain** the purpose of using the circular lists and doubly linked lists.(L2) | |  |
| 6. | **Describe** Josephus problem.(L2) | |  |
| 7. | **Differentiate** between circular singly linked list and circular doubly linked list (L4) | |  |
| 8. | **Describe** the usage of header nodes with lists.(L2) | |  |
| 9. | **Define** circular doubly linked list.(L1) | |  |
| 10. | **Write** programs to add two integers using doubly linked list.(L6) | |  |

# **UNIT 5**

# **MODULE WISE PLAN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Course Code & Title: :* ***3*CCI02, Data Structures** | | | | |
| *Chapter Number & Title:* **Non- Linear Data Structures: Trees, Binary Search Trees, Threaded binary trees** | | *Planned Hours:* **10L+03T Hrs** | | |
| ***Learning Objectives*** | | |  |
| At the end of the chapter the student should be able to: | | |  |
| 1. | **Define** the following (L1)   1. Binary tree. 2. All most complete binary tree | |  |
| 2. | **Explain** the concept of representation of both array and linked list representation of trees. (L2) | |  |
| 3. | **Write** program for creation of binary search tree. (L4) | |  |
| 4. | **Describe** the benefits of using the trees and tree traversal techniques.(L3) | |  |
| 5. | **Design** the heap trees and expression trees.(L3) | |  |
| 6. | **Explain** the in-threaded binary tree with examples.(L2) | |  |

**Faculty Head of the Department Principal**