# Lesson Plan

*Cover Page*: Course Overview

*Semester:* **III**  Year: **2017-18**

|  |  |
| --- | --- |
| *Course Title*:  **DATA STRUCTURES WITH C – THEORY AND LAB** | *Course Code*: **16CS33** |
| *Total Contact Hours*: 36 hrs | *Duration of SEE*: 3 **hrs** |
| *SEE Marks*: **100 + 50** | *CIE Marks*: **100** |
| *Lesson Plan Author*: Dr. Deepamala.N, Prof. Jyothi Shetty, Prof. Girish Rao, Prof. Manas | *Date*: **Jun 06, 2017** |
| *Checked By:* | *Date*: |

## Course Overview:

This course provides students advanced C concepts and its applications. Students can visualize data structures and apply the correct data structure based on requirement.

## Course Learning Objectives-CLO

## Data Structures is a fundamental subject which provides exposure to advanced programming. It gives an idea on effective implementation of programs for an application by visualizing the correct data structure. This course will enlighten students for the future effective programming with complexity taken into consideration. This course lays down the following objectives -

1. Learn the fundamental data structures and identify data structuring strategies that are appropriate to a given contextual problem and able to design, develop, test and debug in C language considering appropriate data structure.
2. Illustrate and implement data types such as stack, queue and linked list and apply them for the given problem.
3. Understand and distinguish the conceptual and applicative differences in trees, binary trees, binary search trees, AVL and splay trees. Apply the correct tree for the given application.
4. Create and use appropriate data structures in C programs for solving real life problems.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Course Outcomes: After completing the course, the students will be able to** | | |  | Understand and explore the fundamental concepts of various data structures. | |  | Analyze and represent various data structures. | |  | Design algorithms on different data structures like Stack, Queue, List, Tree and hashing. | |  | Implement programs with suitable data structure based on the requirements of the application. | |

**Course Content**

**Course Code: 16CS33**

**Hrs/Week L-T-P-S: 3 :0 :1 :1 CIE: 100 marks**

**Teaching Hours: 36 Hrs SEE: 100 marks**

|  |  |
| --- | --- |
| All the data structures has to be implemented using structures  **Unit-I**  **Introduction**  Types of Data Structures, Linear & non-linear Data Structures  **Stacks**  Stack definitions & concepts, Representing stacks in C, Operations on stacks, Applications of Stacks: Infix to Postfix, Infix to Prefix, Postfix expression evaluation  **Recursion**  Introduction to Recursion, Factorial function, Binary search, Towers of Hanoi problem, Role of the stack during execution. | 7 Hours |
| **Unit-II**  **Queues**  Representation of queue, operations, circular queues. Application of Queue: Message queue using circular queue.  **Dynamic Memory allocation:** malloc(), calloc(),free(), realloc() | 7 Hours |
| **Unit-III**  **Linked Lists**  Inserting and removing nodes from a list, getnode and freenode operations, Implementation(insertion, deletion and display) of single Linked list, Double linked list, circular linked list and header nodes.Application of lists: Polynomial multiplication using single linked list, addition of long positive integers using circular single linked list. | 7 Hours |
| **Unit –IV**  **Trees**  Implementation (Insertion, deletion and display) of Binary Trees, Binary search trees (BST), AVL trees, splay trees and Tries. Application of tree: expression trees, Infix, Postfix and Prefix traversals. | 7 Hours |
| **Unit-V**  **Heap**  Heap construction, deletion, Implementation of priority queue.  **Hashing**  Collision concept, Implementation (Insertion and deletion) using Linear Probing, separate chaining, quadratic probing, double hashing. | 8 Hours |
| **Laboratory Component:** | | |
| **PART-A**   |  |  | | --- | --- | | 1 | Use Stack operations to do the following:   1. Assign to a variable name Y the value of the third element from the top of the stack and keep the stack undisturbed. 2. Given an arbitrary integer n pop out the top n elements. A message should be displayed if an unusual condition is encountered. 3. Assign to a variable name Y the value of the third element from the bottom of the stack and keep the stack undisturbed.   (Hint: you may use a temporary stack) | | 2 | Write a C program that parses Infix arithmetic expressions to Postfix arithmetic expressions using a Stack. | | 3 | Write a C program to simulate the working of Messaging System in which a message is placed in a circular Queue by a Message Sender, a message is removed from the circular queue by a Message Receiver, which can also display the contents of the Queue. | | 4 | Implement a program to multiply two polynomials using single linked list. | | 5 | Write a C program to implement addition of long positive integers using circular single linked list with header node. | | 6 | Design a doubly linked list to represent sparse matrix. Each node in the list can have the row and column index of the matrix element and the value of the element. Print the complete matrix as the output. | | 7 | Write a C program to create Binary Tree and provide insertion and deletion operations and to traverse the tree using In-order, Preorder and Post order (recursively) | | 8 | Given a String representing a parentheses-free infix arithmetic expression, implement a program to place it in a tree in the infix form. Assume that a variable name is a single letter. Traverse the tree to produce an equivalent postfix and prefix expression string. | | 9 | Write a C program to implement Hashing using Linear probing. Implement insertion, deletion, search and display. | | 10 | Write a C program to implement priority queue to insert, delete and display the elements. | | **PART – B** | | | Student will design, develop and implement an application using the appropriate data structure. Some example applications are listed below:   * Huffman coding * Dictionary implementation for Indian Languages * Stemmer implementation for Indian language * Word frequency finder. * Bitmap Image Compression. * Binary Tree (Graphical Implementation) * To store a set of programs which are to be given access to a hard disk according to their priority * For representing a city region telephone network. * To store a set of fixed key words which are referenced very frequently. * To represent an image in the form of a bitmap. * To implement back functionality in the internet browser. * To store dynamically growing data which is accessed very frequently, based upon a key value. * To implement printer spooler so that jobs can be printed in the order of their arrival. * To record the sequence of all the pages browsed in one session. * To implement the undo function. * To store information about the directories and files in a system. | | | | |

**REFERENCE BOOKS:**

1. Yedidyah Langsam Moshe J. Augenstein and Aaron M. Tenenbaum, ”Data Structures

using C and C++”, PHI/Pearson, 2nd Edition, 2009.

1. Mark Allen Weiss,“Data Structures and Algorithm Analysis in C++”, Addison-Wesley; 4th Revised edition; 2013, ISBN-13**:** 9780132847377
2. ReemaThareja,“Data Structures Using C”, Oxford Higher Education , First Edition, 2011.
3. Behrouz A. Forouzan and Richard F. Gilberg,“Computer Science A Structured Programming Approach Using C”, 3rd Edition, Thomson, 2009.
4. R.Kruse, C.L Tondo and B.Leung, “Data Structures and Program Design in C++”, 2nd Edition, Pearson Education, 2009.
5. Ashok N Kamthane, “Introduction to Data Structures in C”, Pearson Education, 1st Edition, 2009.
6. Pierre Bourque, Richard E. Fairley, “Sweebok: Guide to the software engineering body of knowledge”, Version 3, IEEE society project.

**Unit and Chapter wise Plan**

**Unit 1**

|  |  |
| --- | --- |
| *Course Code and Title:* (**16CS33) Data Structures using C** | |
| *Chapter Number and Title*:1. Introduction, Stacks, Recursion | *Planned Hours:* ***7* hrs** |

## Learning Objectives:

|  |  |
| --- | --- |
| **Sl. No.** | **Objectives** |
| 1 | Introduction to Types of Data Structures, Linear & non-linear Data Structures |
| 2 | Introduction to stack, stack operations : Push and Pop, Applications of stack. |
| 3 | Recursion and its applications |

|  |
| --- |
| ***Lesson Schedule***  *Class No. Portion covered per hour*   1. Types of Data Structures, Linear & non linear Data Structures 2. Stack definitions & concepts, Representing stacks in C 3. Operations on stacks, Applications of Stacks, Infix to Postfix 4. Infix to Prefix 5. Postfix expression evaluation 6. Recursion 7. Applications of recursion |

## Model Questions

1. List and explain various types of Data Structures
2. Develop step by step algorithm or(C-program) to convert a given infix expression to postfix expression.
3. Apply the above algorithm to convert the following infix expression to postfix expression. Q: A+B\*C-(D/E^F)\*G)\*H.
4. Write an algorithm for evaluating a valid postfix expression. Trace the same on AB+C-BA+C$- for given value A=1,B=2 and C=3.
5. List applications of stacks. Using stack write an algorithm to determine if a given string is palindrome or not.
6. What is recursion? Write recursive program to for “Tower of Hanoi”
7. Define a recursion. List the important features of a recursive program.
8. Compare iteration with that of recursion.Define with atleast three parameters: recursion relation for GCD of two numbers and formulate a recursive procedure in C for the same.
9. Compare and contrast the recursive and iterative methods in Binary Search.

**UNIT-II**

|  |  |
| --- | --- |
| *Course Code and Title:* (**16CS33) Data Structures using C** | |
| *Chapter Number and Title*: *2****.***   **Queues, Dynamic Memory allocation** | *Planned Hours:* ***07*  hrs** |

## Learning Objectives

|  |  |
| --- | --- |
| **Sl. No.** | **Objectives** |
| **1** | Understanding of The queue and its sequential representation. |
| **2** | C implementation of linear queues, Insert operation, and delete operation. |
| **3** | Concept of priority queue, Array implementation of a priority queue. |
| **4** | Implementation of priority queue using binary heap |
| **5** | Difference between static memory allocation and dynamic allocation |

|  |
| --- |
| ***Lesson Schedule***  *Class No. Portion covered per hour*   1. Representation of queue 2. Operations on queue 3. Circular queues 4. Implementation of Priority queues 5. Message queue using circular queue 6. Applications of queues 7. Dynamic Memory allocation:   malloc(), calloc(), free(), realloc() |

**Model Questions**

1. What is circular queue? write the implementation of circular queue using array.
2. Give a complete specification of data structure QUEUE.
3. Differentiate between linear and circular queue
4. Explain array implementation of priority queue.
5. What is binary heap? Explain the implementation of ascending priority queue using MIN Heap.
6. With syntax give an example for dynamic memory allocation functions

**Unit III**

|  |  |
| --- | --- |
| *Course Code and Title:* (**16CS33) Data Structures using C** | |
| *Chapter Number and Title*: *3*  **Linked Lists** | *Planned Hours:* ***07* hrs** |

## Learning Objectives

|  |  |
| --- | --- |
| **Sl. No.** | **Objectives** |
| **1** | Understanding of Linked lists: Inserting and removing nodes from a list |
| **2** | Define getnode and freenode operations, Linked implementation of singly list |
| **3** | Example of list operations. |
| **4** | Lists in C: Array implementation of lists, Limitations of array implementation, Allocating and freeing dynamic variables |
| **5** | Linked list as using dynamic variable |

|  |
| --- |
| ***Lesson Schedule***  *Class No. Portion covered per hour*   1. Inserting and removing nodes from a list 2. Linked implementation of Stacks, Queues 3. getnode and freenode operations 4. Example of list operations 5. Array implementation 6. Limitations of the array implementation 7. Linked list using dynamic variables and its operations. (multiplication of polynomials and addition of long integers) |

**Model Questions**

1. Write routines in C for performing the following:
   1. to find the sum of all the elements in a singly linked list
   2. to find the average of a set of elements in a singly linked list.
2. An ITEM of information is to be inserted in a sorted linked list. Develop an algorithm or C-routine for the given task.
3. With an example discuss how the data structure linked list is represented in memory.
4. A circular queue, size of which is 5 has three elements 20,40 and 60,where front=0 and rear=2.Show with necessary diagrams what is the value of front and rear after each of these operations: i)insert item 50 ii)insert item 10

iii)insert item 30 iv)delete an item v)delete an item.

1. What is circular queue? Write the implementation of circular queue using linked list.
2. What is the advantage of representing a group of integers using linked list?
3. Write routines in C for performing the following:
   1. to find the sum of all the elements in a doubly linked list
   2. to find the average of a set of elements in a doubly linked list.
4. Design algorithm to create circular list of long positive integers.
5. A circular list, size of which is 5 has three elements 20,40 and 60,where front=0 and rear=2.Show with necessary diagrams what is the value of front and rear after each of these operations: i)insert item 50 ii)insert item 10
   * 1. iii)insert item 30 iv)delete an item v)delete an item.
6. Represent a stack using linked list.

**UNIT -IV**

|  |  |
| --- | --- |
| *Course Code and Title:* (**16CS33) Data Structures using C** | |
| *Chapter Number and Title*:4.  **Trees** | *Planned Hours:* ***07* hrs** |

## Learning Objectives

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | | **Objectives** | |
| **1** | | Understanding binary trees and its types | |
| **2** | | Learning Binary Search Tree (BST), Insertion and Deletion in BST. | |
| **3** | | Know how to Expression trees, Infix, Postfix, and Prefix Traversals, Splay tree, Tries. | |
| **4** | | Understanding complete binary almost complete binary tree | |
| **5** | | Know implementation of searching technique using binary trees | |
| **6** | | Understanding binary trees and its types | |
| **7** | | Learning Binary Search Tree (BST), Insertion and Deletion in BST. | |
| ***Lesson Schedule***  *Class No. Portion covered per hour*   1. Trees and Binary Trees 2. Binary Search Tree (BST) 3. Insertion and Deletion in BST 4. AVL Trees 5. Splay trees 6. Tries 7. Expression trees, Infix, Postfix, and Prefix Traversals, | |

**Model Questions**

1. What is binary tree? Illustrate example for complete & almost binary tree
2. Write pseudo code for Binary search tree.
3. Give an example for traversal of expression in following methods

i)Infix ii) prefix iii) postfix

1. What are splay trees? Explain with example.
2. What are operation performed on binary search trees? List & explain.

## UNIT- V

|  |  |
| --- | --- |
| *Course Code and Title:* (**16CS33) Data Structure using C** | |
| *Chapter Number and Title*: *5****.***  Heap, Hashing | *Planned Hours:* ***08* hrs** |

## Learning Objectives

|  |  |
| --- | --- |
| **Sl. No.** | **Objectives** |
| **1** | **Introduction to Heap** |
| **2** | Insertion, deletion into a heap |
| **3** | Implementation of priority queue |
| **4** | Collision in hasining, Insertion, deletion using linear probing, separate chaining. |
| **5** | Exposure to separate chaining, quadratic probing and double hasining. |

|  |
| --- |
| ***Lesson Schedule***  *Class No. Portion covered per hour*   1. Introduction to Heap, Insertion and deletion 2. Implementation of priority queue 3. Hashing introduction and collision 4. Linear probing 5. Separate chaining 6. Quadratic probing 7. Double hashing 8. Other applications of hashing |

**Model Questions**

1. What is a heap? Write a program to implement priority queue.
2. What is hashing and list the types of hashing?
3. Explain Linear probing with an example
4. Differentiate between quadratic probing and double hashing with an example

**LESSON PLAN /WEEK**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No** | **Topic** | **Sub Topic** | **No. of Hrs** | **Activity & Materials to be used** | **Book Referred** |
| 1 | Introduction, Stacks, Recursion. | Types of Data Structures, | 0.5 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Linear & non linear Data Structures | 0.5 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Stack definitions & concepts | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Representing stacks in C | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Operations on stacks | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Applications of Stacks, Infix to Postfix | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Postfix expression evaluation | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Introduction to Recursion, Factorial function, Binary search, Towers of Hanoi problem, Role of the stack during execution. | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| 2 | Queue | Representation of queue, | 2 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| operations, circular queues | 2 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Application of Queue: Message queue using circular queue. | 2 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Dynamic Memory allocation | malloc(), calloc(),  free(), realloc()  Practice Programs | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| 3 | Linked Lists | Inserting and removing nodes from a list | 2 | Flipped classroom with Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| linked implementation of Stacks, Queues. getnode and freenode operations | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Double linked list, circular linked list and header nodes | 2 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| .Application of lists: Polynomial multiplication using single linked list, addition of long positive integers using circular single linked list. | 2 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| 4 | Trees | Implementation (Insertion, deletion and display) of Binary Trees, | 2 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Binary search trees (BST) | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| AVL | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Splay Trees and Tries | 2 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| Application of tree: expression trees, Infix, Postfix ad Prefix traversals. | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
| 5 | Heap, Hashing | Heap construction, deletion | 2 | Lecture(Black Board) with TPS | Aaron M. Tenenbaum; Data Structures using C and C++ |
|  | Implementation of priority queue. | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
|  | Hashing | 1 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
|  | Implementation (Insertion and deletion) using Linear Probing | 1 | Flipped classroom + Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |
|  | separate chaining, quadratic probing, double hashing. | 2 | Lecture(Black Board) | Aaron M. Tenenbaum; Data Structures using C and C++ |

**Lab Programs**

|  |  |
| --- | --- |
| 1 | Use Stack operations to do the following:   1. Assign to a variable name Y the value of the third element from the top of the stack and keep the stack undisturbed. 2. Given an arbitrary integer n pop out the top n elements. A message should be displayed if an unusual condition is encountered. 3. Assign to a variable name Y the value of the third element from the bottom of the stack and keep the stack undisturbed.   (Hint: you may use a temporary stack) |
| 2 | Write a C program that parses Infix arithmetic expressions to Postfix arithmetic expressions using a Stack. |
| 3 | Write a C program to simulate the working of Messaging System in which a message is placed in a circular Queue by a Message Sender, a message is removed from the circular queue by a Message Receiver, which can also display the contents of the Queue. |
| 4 | Implement a program to multiply two polynomials using single linked list. |
| 5 | Write a C program to implement addition of long positive integers using circular single linked list with header node. |
| 6 | Design a doubly linked list to represent sparse matrix. Each node in the list can have the row and column index of the matrix element and the value of the element. Print the complete matrix as the output. |
| 7 | Write a C program to create Binary Tree and provide insertion and deletion operations and to traverse the tree using In-order, Preorder and Post order (recursively) |
| 8 | Given a String representing a parentheses-free infix arithmetic expression, implement a program to place it in a tree in the infix form. Assume that a variable name is a single letter. Traverse the tree to produce an equivalent postfix and prefix expression string. |
| 9 | Write a C program to implement Hashing using Linear probing. Implement insertion, deletion, search and display. |
| 10 | Write a C program to implement priority queue to insert, delete and display the elements. |
| **PART – B** | |
| Student will design, develop and implement an application using the appropriate data structure. Some example applications are listed below:   * Huffman coding * Dictionary implementation for Indian Languages * Stemmer implementation for Indian language * Word frequency finder. * Bitmap Image Compression. * Binary Tree (Graphical Implementation) * To store a set of programs which are to be given access to a hard disk according to their priority * For representing a city region telephone network. * To store a set of fixed key words which are referenced very frequently. * To represent an image in the form of a bitmap. * To implement back functionality in the internet browser. * To store dynamically growing data which is accessed very frequently, based upon a key value. * To implement printer spooler so that jobs can be printed in the order of their arrival. * To record the sequence of all the pages browsed in one session. * To implement the undo function. * To store information about the directories and files in a system. | |

**Each program is evaluated for 10 marks.**

**Lab Write-up & Execution rubrics (Max: 6 marks) and Viva Voce rubrics (Max: 4 marks)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sl. No** | **Criteria** | **Measuring methods** | **Excellent** | **Good** | **Poor** |  |
| 1 | **Understanding of problem statement and design.**  **(2 Marks)**  **CO1,CO2** | Observations | Student exhibits thorough understanding of requirements and applies suitable logic and programming concepts for the problem  **(2 M)** | Student has sufficient understanding of requirements and applies suitable logic and programming concepts for the problem.  **(<2 M and >=1 M)** | Student does not have a clear understanding of requirements and is unable to apply suitable logic and programming concept for the problem.    **(0 M)** |  |
| 2 | **Execution and debugging**  **(2 Marks)**    **CO4** | Observations | Student demonstrates the execution of the program with efficient code. Appropriate validations with all test cases are handled.  **(2 M)** | Student demonstrates the execution of the program without efficiency of the code and validates only few cases.      **(1 M)** | Student has not executed the program.    **(0 M)** |  |
| 3 | **Results and Documentation**  **(2 Marks)**  **CO2** | Observations | Documentation with appropriate comments and output is covered in data sheets and manual.  **(2 M)** | Documentation with only few comments and only few output cases is covered in data sheets and manual.  **(1 M)** | Documentation with no comments and no output cases is covered in data sheets and manual.  **(0 M)** |  |
| **Viva Voce rubrics (Max: 4 marks)** | | | | | |  |
| 1 | **Conceptual Understanding**  **(2 Marks)**  **CO1** | Viva Voce | Explains thoroughly the programming concepts and the data structure.  **(2 M)** | Adequately explains the programming concepts and data structures.    **(1 M)** | Unable to explain the programming concepts and data structure.      **(0 M)** |  |
| 2 | **Use of data structure**  **(1 Marks)**  **CO4** | Viva Voce | Insightful explanation of data structure design technique for the given problem to derive solution.  **(1 M)** | Sufficiently explains the use of appropriate data structure design technique for the given problem to derive solution.  **(0.5 M)** | Unable to explain the data structure design technique for the given problem.  **(0 M)** |  |
| 3 | **Communication of Concepts**  **(1 Marks)**  **CO4** | Viva Voce | Communicates the concept used in problem solving well.  **(1 M)** | Sufficiently communicates the concepts used in problem solving.  **(0.5 M)** | Unable to communicate the concepts used in problem.  **(0 M)** |  |

**Part B program will be evaluated for 10 marks and rubrics is :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Components** |  | | | **Marks Allocated** |
|  | **Excellent** | **Good** | **Not Satisfactory** |  |
| **Problem definition(1)** | Exhibit Clear understanding of the problem and requires various data structures to bring out the solution  1 marks | Exhibit understanding of the program but requires simple data structures to bring out the solution.  < 1 marks | Not Clear about the program- 0 Marks |  |
| **Application of relevant data structure with justification**  **(1)** | Relevant Data Structure used with correct justification  1 marks | Not proper data structure used and improper justification  <1 marks | No justification 0 marks |  |
| **Incorporation of suggestions (1)** | Changes are made as per modification suggested during evaluation and new innovations added.  (1) | Changes are made as per modifications suggested during evaluation and good justification.  <1 marks | Suggested changes not incorporated  0 marks |  |
| **Design (1)** | Analyze and draw a system diagram for the problem  1 marks | Diagram not proper  <1 marks | No design  0 |  |
| **Source Code (2)** | Coded with proper coding guide lines and all the cases handled (3) | Coded with coding guidelines but does not cover all the cases  <3 and >=1 | Coded without any coding guide lines and does not handle all the cases  <1 marks |  |
| **Demonstration of Output (2)** | Demonstrated correct output satisfying all the cases. –2marks | Demonstrated partial output. - <=1 | Not clear output. -0 marks |  |
| **Documentation (1)** | Correctly documented with all the result – 2 marks | Partial documentation - <=1 marks | No documentation – 0 marks |  |
| **Total** |  | | |  |
| **Signature of the Staff** |  | |  | |

# Evaluation Scheme

|  |  |
| --- | --- |
| **Continuous Internal Evaluation (CIE)**  **( Theory – 100 Marks)** | |
| Evaluation method | Course with Self-study |
| Quiz -1 | 10 |
| Test -1 | 25 |
| Quiz -2 | 10 |
| Quiz -3 | 10 |
| Test -2 | 25 |
| Self-study (EL) | 10 |
| Active Learning | 10 |
| **Total – theory** | **100** |
| **Lab** | **50** |
| **Total** | **150** |

**Self-Study or Experiential Learning**

**The course has experiential learning as one of the assessment tool. The topic chosen in Part – B for Lab is evaluated in EL also. In this students are assessed based on following Components**

Experiential learning Rubrics

|  |  |  |  |
| --- | --- | --- | --- |
| **Introduction**  **(2 Marks)** | Detailed and  extensive  explanation  of the purpose and  uniqueness of the project  ( 2Marks) | Good  Explanation of the  purpose and relation with the program of study and usefulness  (<2 and >0) | Minimal  explanation of  the purpose and relation with the program of study  (0 Marks) |
| **Objectives**  **and**  **Methodology**  **of the**  **Proposed**  **Work**  **(2 Marks)** | All objectives of the proposed work are well  defined; Steps to be  followed to solve the  defined problem are  clearly specified with consideration to environment, safety and ethics.  (2 Marks) | Good justification  to the objectives;  Methodology to be  followed is specified but detailing is  not done. Objectives with consideration to environment, safety and ethics is defined.  (< 2 Marks and >0 Marks) | Objectives of  the proposed  work are  either not  identified or  not well  defined;  Incomplete  and improper  identification of objectives w.r.t environment, safety and ethics.  (0 Marks) |
| **Implementation**  **(2 Marks)** | The implementation consistent with problem statement, coding standards and optimized. Clear and understandable description of the functional status of each features or modules.  (2 Marks) | The implementation consistent with problem statement, understandable but not clearly defined  (<2 and >0) | Completely different from definition.  (0 Marks) |
| **Demo of the project**  **(2 Marks)** | Demo shown and complete  ( 2 marks) | Demo shown but incomplete  (<2 and >0) | No demo  (0 marks) |
| **Presentation**  **(2 Marks)** | Preparations of the presenters and Response of presenters for questions and critique  ( 2 Marks) | Preparations of the presenters not satisfactory but Response of presenters for questions and critique satisfactory  (<2 and > 0) | Irrelevant presentation  (0 Marks) |

|  |  |
| --- | --- |
| **Semester End Evaluation Theory (100)** | |
| **Part- –A**  **Objective type questions** | **20** |
| **Part –B**  There should be five questions from five units. Each question should be for maximum of 16 Marks.  The **UNIT-1**, **UNIT-4** and **UNIT-5** should not have any choice.  The **UNIT-2 and UNIT-3** should have an internal choice.  Both the questions should be of the same complexity in terms of COs and Bloom’s taxonomy level. | **80** |
|
|
| **Total** | **100** |

## Course Unitization for Internals and Semester End Examination

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Part** | **Chapter** | | **Teaching Hours** | **No. of Questions in** | | **No. of Questions in SEE** |
| **Internals I** | **Internals II** |
| **Unit 1** | 1 | Introduction, Stacks | 5 | 2 | -- | 2 |
| 2 | **Recursion** | 2 | 1 | -- |
| **Unit 2** | 3 | **Queues** | 6 | 2 | -- | 1 |
| 4 | **Dynamic memory allocation** | 1 | 1 | -- | 1 |
| **Unit 3** | 5 | **Linked Lists** | 5 |  | 2 | 2 |
| 6 | **Applications of linkedlist** | 2 | -- | 1 | 1 |
| **Unit 4** | 7 | **Trees** | 5 | -- | 1 | 1 |
| 8 | **Applications of trees** | 2 | -- | 1 |
| **Unit 5** | 9 | **Heaps** | 5 | -- | 1 | 1 |
| 10 | **Hashing** | 3 | -- | 1 |

**Faculties In-charge Head of Department**

**Course Articulation Matrix (16CS33)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Outcomes Vs Program Outcome (16CS33)** | | | | | | | | | | | | |
|  | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| CO1 | L | H | H | L | L | L | - | - | - | - | - | M |
| CO2 | L | M | M | L | L | L | - | - | - | - | - | M |
| CO3 | L | M | M | L | L | L | - | - | - | - | - | M |
| CO4 | L | H | H | L | L | L | - | - | - | - | - | M |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Program Articulation Matrix**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Course - PO Mapping** | | | | | | | | | | | | | | |  | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | | **Course** | L | H | H | L | L | L | - | - | - | - | - | M | |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **CO –PSO Mapping** | | |  | **Course – PSO Mapping** | | | | **CO/PSO** | **PSO1** | **PSO2** |  | **Course** | **PSO1** | **PSO2** | | **CO1** | M | M |  | **M** | **M** | | **CO2** | M | M |  |  |  |  | | **CO3** | M | M |  |  |  |  | | **CO4** | M | M |  |  |  |  |   **CO-PSO Mapping COURSE-PSO Mapping** |

**Active Learning**

**Evaluation is for 10 Marks**

**Conduct tests using tool like Hackerrank**

|  |  |  |
| --- | --- | --- |
| **Test** | **Syllabus** | **Marks** |
| **Test 1** | **Pointers** | **3 Marks** |
| **Test 2** | **Unit 1,2** | **4 Marks** |
| **Test 3** | **Unit 3,4** | **3 Marks** |
| **Test 4** | **Unit 5** | **Optional** |

**Flipped Classes**

**A video on Pointers will be shared with students which will be evaluated as part of Test 1 in Active learning.**

**A video on applications of linked list and trees will be shared which will be evaluated as part of Active learning.**

**QEEE classes will also be conducted as part of flipped classes**

1. **Video on Pointers**

[**https://drive.google.com/open?id=0BwdO-ntZaW5CeUtwRE5zTElGbmc**](https://drive.google.com/open?id=0BwdO-ntZaW5CeUtwRE5zTElGbmc)

1. **Evaluation postfix expression video link**

<https://youtu.be/htBf6gX8VNw>

1. **Recursion Videos**

[**https://youtu.be/gNShlZ5xftE**](https://youtu.be/gNShlZ5xftE)

[**https://youtu.be/ZD7zev7Lt70**](https://youtu.be/ZD7zev7Lt70)

**DSC Activity Incharges:**

|  |  |  |
| --- | --- | --- |
| **Sl No** | **Topic** | **Faculty** |
| 1 | Lab Incharge | Dr. Deepamala.N |
| 2 | Self-Study or EL | Prof. Jyothi Shetty, Dr. Deepamala.N, Prof. Girish Rao |
| 3 | QEEE | Prof. Manas |

|  |  |  |
| --- | --- | --- |
| **Sl No** | **Topic** | **Faculty** |
| 1 | Recursion and its Applications | Dr. Deepamala.N |
| 2 | Evaluation of Postfix Expression | Prof. Jyoti Shetty |
| 3 | Pointers | Prof. Manas |