# Computer Programming Laboratory

**B.Tech: lll Semester**

**Department: Computer Science and Engineering**

**Faculty of Engineering & Technology**
**Ramaiah University of Applied Sciences**

| | |
|---|---|
| Faculty | Engineering & Technology |
| Programme | B. Tech. |
| Course | Computer Science and Engineering |
| Year/Semester | 2nd Year/3rd Semester |
| Name of the Laboratory | Data structures and Algorithms Laboratory |
| Laboratory Code | CSC202A |

List of Experiments

1. Data types, local variables and Random number generation
2. Using dynamic memory, Arrays, Vectors and String pattern matching
3. Test case design, Linked lists, deques
4. Stacks and queues
5. Binary Trees
6. Algorithms on Binary Trees
7. Graphs
8. Algorithms on Graphs
9. Algorithms on Graphs
10. Heaps, Comparator, Priority Queues and sorting algorithms
11. Search algorithms
12. Hashing and Hash tables
13. External Storage and related algorithms
14. Sets and Partitions
15. Advanced tree data structures
16. Advanced queue data structures
17. Advanced heap data structures

## Laboratory 1

Title of the Laboratory Exercise: Data types, local variables and Random number generation

1. Introduction and Purpose of Experiment

   Students get familiar with the data types and local variables and random number generation. Basic concepts such as data types and local variables are part and parcel of almost all the c programs. Hence sound knowledge is most essential in this regard. Also, the random number generation essential for many applications, for ex. rolling a dice for many in gaming applications such as backgammon which requires a random number generation from 1 to 6.

2. Aim and Objectives

   Aim

   To design and develop a C programs using Data types, local variables and Random number generation to demonstrate the use and significate of the same in programming.

   Objectives

   At the end of this lab, the student will be able to

   - Use variables of the basic data types with proper declarations
   - Read and validate the input data
   - Generate random numbers for any application

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C language
   iv. Compile the C program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

4. Questions

   Demonstrate the use of data types, local variables and Random numbers by designing appropriate algorithms for the below problems. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of the algorithm. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome.

   i. Write a C program to illustrate random number generation. Modify the program to generate a random number between 75 to 85
   ii. Write a C program to find sum of n elements, allocate memory dynamically using malloc() and calloc() function. Modify the program include both the allocation strategies in a single program.
   iii. Combine both random number generation and memory allocation in a single program to demonstrate the allocation of random number of memory blocks.

5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

   1. Limitations of Experiments

   2. Limitations of Results

   3. Learning happened

4. Recommendations

## Laboratory 2

Title of the Laboratory Exercise: Using dynamic memory, Arrays, Vectors and String pattern matching

1. Introduction and Purpose of Experiment

   Dynamic memory allocation is an important concept in C for allocation and release of the allocated memory dynamically which would help in many applications. This experiment includes creating and applying a vector ADT which uses the dynamic memory allocation concept. Use the created vector ADT for string pattern matching.

2. Aim and Objectives

   Aim

   To develop vector ADT and to use vector ADT for string pattern matching also demonstrate the concept of dynamic memory allocation

   Objectives

   At the end of this lab, the student will be able to

   - Illustrate the concept of dynamic memory allocation
   - Create vector ADT
   - Apply vector ADT for string pattern matching application

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C language
   iv. Compile the C program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

4. Questions

Design the vector ADT and apply vector ADT for string matching problem. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of the algorithm designed. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome.

5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

   1. Limitations of Experiments

   2. Limitations of Results

   3. Learning happened

   4. Recommendations

## Laboratory 3

Title of the Laboratory Exercise: Test case design, Linked lists, deques

1. Introduction and Purpose of Experiment

   Testing improves the quality and reliability of a 'C' program. This experiment introduces test case design for a 'C' program to test the desired program for bugs and errors. This experiment also includes design and implementation of algorithms for creating and applying linked list and deques. The test cases are written to validate the design and implementation of linked list ADT.

2. Aim and Objectives

   Aim

   To design and develop linked list ADT and deque ADT and design test cases to test the both the programs.

   Objectives

   At the end of this lab, the student will be able to

   - Design and develop test cases for a given C program
   - Design and develop linked list and deque ADT
   - Application of ADT to illustrate the addition and deletion of the elements in the middle of the collection.

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C language
   iv. Compile the C program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

4. Questions

   Design the data structures, algorithm and write the program to implement a linked list ADT and deque ADT. Apply the designed ADT to demonstrate its operations on integer data and perform addition and deletion, insertion operation. Design the test cases and validate the

results. Analyse the efficiency of the algorithm. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome.

5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

    1. Limitations of Experiments

    2. Limitations of Results

    3. Learning happened

    4. Recommendations

## Laboratory 4

Title of the Laboratory Exercise: Stacks and queues

Introduction and Purpose of Experiment

Stacks and queues are very important data structures used in many real time applications. This experiment introduces the development of stack and queue ADT and applying them together.

1. Aim and Objectives

   Aim

   - To develop stack and queue ADT and to use them for string applications

   Objectives

   At the end of this lab, the student will be able to

   - Design and develop and use stack and demonstrate its operations
   - Design and develop and use queue and demonstrate its operations

2. Experimental Procedure
   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C language
   iv. Compile the C program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment
3. Questions

   Design Stack and Queue data structure ADT using linked list, apply them to check whether a given string is palindrome or not. Write an algorithm and implement the same. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of the algorithm. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome.

4. Calculations/Computations/Algorithms

5. Presentation of Results

6. Analysis and Discussions

7. Conclusions

8. Comments

    1. Limitations of Experiments

    2. Limitations of Results

    3. Learning happened

    4. Recommendations

## Laboratory 5

Title of the Laboratory Exercise: Binary trees

1. Introduction and Purpose of Experiment

   Linear organization used on arrays, vectors, stacks and queues become inefficient in some applications. Then we choose the structures which provide non-linear organization. Binary tree is a non-linear data structure used in many applications. This experiment introduces binary search trees and its applications.

2. Aim and Objectives

Aim

- To develop Binary search tree ADT

Objectives

At the end of this lab, the student will be able to

- Design binary tree ADT
- Use binary tree ADT to illustrate the binary tree operations
- Use binary tree ADT and illustrate binary tree traversals

3. Experimental Procedure

    i. Analyse the problem statement
    ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
    iii. Implement the algorithm in C language
    iv. Compile the C program
    v. Test the implemented program
    vi. Document the Results
    vii. Analyse and discuss the outcomes of your experiment

4. Questions

    Design the Binary tree ADT and write the program and demonstrate its operations such as insert, delete and traversal operations on integer data. Produce output for binary tree traversals. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of the algorithm. Describe your learning along with the limitations of overall approach if any. Suggest how these can be overcome.

5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

    1. Limitations of Experiments

    2. Limitations of Results

    3. Learning happened

    4. Recommendations

## Laboratory 6

Title of the Laboratory Exercise: Algorithms on binary trees

1. Introduction and Purpose of Experiment

   Binary tree algorithms such as Breadth First Search (BFS), Depth First Search (DFS) are useful in many applications like expression tree, decision tree etc. this experiment introduces the binary search algorithm and its applications.

   Aim and Objectives

   Aim

   - To develop programs to implement binary search tree algorithms and demonstrate its traversals such as DFS and BFS.

   Objectives

At the end of this lab, the student will be able to

- Design and Implement BFS tree algorithm
- Apply BFS for traversal and search
- Design and Implement DFS tree algorithm
- Apply DFS for traversal and search

2. Experimental Procedure

    i. Analyse the problem statement
    ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
    iii. Implement the algorithm in C language
    iv. Compile the C program
    v. Test the implemented program
    vi. Document the Results
    vii. Analyse and discuss the outcomes of your experiment

3. Questions

    Design and develop DFS and BFS algorithms for a given binary search tree and write a program to search a node in a tree. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of both the algorithms. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

4. Calculations/Computations/Algorithms

5. Presentation of Results

6. Analysis and Discussions

7. Conclusions

8. Comments

1. Limitations of Experiments

2. Limitations of Results

3. Learning happened

4. Recommendations

## Laboratory 7

Title of the Laboratory Exercise: Advanced tree data structures

1. Introduction and Purpose of Experiment

   Advanced data structures such as AVL trees help to overcome many limitations of binary trees. In this experiment students will be able to implement and analyse the efficiency of AVL trees and its operations.

   Aim and Objectives

   Aim

   - To design and develop algorithms for creating and demonstrating AVL tree operations

   Objectives

   At the end of this lab, the student will be able to

   - Create AVL tree
   - Demonstrate inserting and deleting a node from AVL tree.

2. Experimental Procedure

   i. Analyse the problem statement

ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
iii. Implement the algorithm in C language
iv. Compile the C program
v. Test the implemented program
vi. Document the Results
vii. Analyse and discuss the outcomes of your experiment

3. Questions

Design an algorithms for creating an AVL tree, inserting and deleting a node from the AVL tree and display its content. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of AVL tree over binary trees. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

4. Calculations/Computations/Algorithms

5. Presentation of Results

6. Analysis and Discussions

7. Conclusions

8. Comments

1. Limitations of Experiments

2. Limitations of Results

3. Learning happened

4. Recommendations

## Laboratory 8

Title of the Laboratory Exercise: Hashing and Hash tables

1. Introduction and Purpose of Experiment

   Hashing is an efficient search technique used in many applications. A **hash table** is a collection of items which are stored in such a way as to make it easy to find them later. In this experiments will learn how to implement and search an element in hash table and analyse the efficiency of hashing technique.

   Aim and Objectives

   Aim

   - To design and develop hash table and demonstrate search operation

   Objectives

   At the end of this lab, the student will be able to

   - Implement hash table
   - Search an element using hash table

2. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C/java language
   iv. Compile the C/java program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

3. Questions

Design an algorithm and write a program to create a hash table and use suitable hash function and demonstrate search operation for a given data. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of hash function used. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

4. Calculations/Computations/Algorithms

5. Presentation of Results

6. Analysis and Discussions

7. Conclusions

8. Comments

   1. Limitations of Experiments

   2. Limitations of Results

   3. Learning happened

   4. Recommendations

## Laboratory 9

Title of the Laboratory Exercise: Graphs

1. Introduction and Purpose of Experiment

   Graphs are important data structures used in many applications like network modelling applications, routing, traffic, water supply etc. This experiment introduces graphs and its representation. A graph can be represented as operations.

   Aim and Objectives

   Aim

   - To design and develop graph ADT and demonstrate its operations

   Objectives

   At the end of this lab, the student will be able to

   - Represent a graph data structure using adjacent list and adjacency list
   - Compare and contrast between both the approaches.

2. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C/java language
   iv. Compile the C program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

3. Questions

   Design an algorithm to represent a graph using adjacency list and adjacency matrix. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of both the algorithms. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

4. Calculations/Computations/Algorithms

5. Presentation of Results

6. Analysis and Discussions

7. Conclusions

8. Comments

    1. Limitations of Experiments

    2. Limitations of Results

    3. Learning happened

    4. Recommendations

## Laboratory 10

Title of the Laboratory Exercise: Graph algorithms

1. Introduction and Purpose of Experiment

Graph algorithms such Depth First Search (DFS) and Breadth First Search (BFS) are very important in many graph applications which are used to traverse the graphs, check for graph connectivity and to find spanning trees etc. This experiment introduces the applications DFS and BFS.

2. Aim and Objectives

Aim

- To apply DFS and BFS algorithm for graph applications

Objectives

At the end of this lab, the student will be able to

- Apply DFS and BFS for graph applications such as graph connectivity check
- Compare the efficiency of both

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C/java language
   iv. Compile the C/java program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of your experiment

4. Questions

Design and develop algorithms to check whether given graph is connected or not using DFS and BFS. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of both the algorithms. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

    1. Limitations of Experiments

    2. Limitations of Results

    3. Learning happened

    4. Recommendations

## Laboratory 11

Title of the Laboratory Exercise:  Sorting Algorithms

1. Introduction and Purpose of Experiment

   Sorting provide us with means of organising information to facilitate the retrieval of specific data. Searching methods are designed to take advantage of the organisation of information. By solving these problems students will be able to use sorting algorithms to sort a randomly ordered set of numbers, and search for key element.

2. Aim and Objectives

   Aim

   - To develop programs for sorting algorithms

   Objectives

At the end of this lab, the student will be able to

- Create C programs using sorting algorithms such as **heap sort**
- Create C programs using sorting algorithms such as **shell sort**

3. Experimental Procedure
    i. Analyse the problem statement
    ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
    iii. Implement the algorithm in C language
    iv. Compile the C program
    v. Test the implemented program
    vi. Document the Results
    vii. Analyse and discuss the outcomes of your experiment

4. Questions
    1) Given a randomly ordered set of n numbers, design and develop an algorithm to sort them into non-descending order using heap sort and shell sort compare their efficiency. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of both the algorithms. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

1. Limitations of Experiments

2. Limitations of Results

3. Learning happened

4. Recommendations

## Laboratory 12

Title of the Laboratory Exercise: Sorting Algorithms

1. Introduction and Purpose of Experiment

   Sorting provide us with means of organising information to facilitate the retrieval of specific data. Searching methods are designed to take advantage of the organisation of information. By solving these problems students will be able to use sorting algorithms to sort a randomly ordered set of numbers, and search for key element.

2. Aim and Objectives

   Aim

   - To develop programs for sorting algorithms

   Objectives

   At the end of this lab, the student will be able to

   - Create C programs using sorting algorithms such as **quick sort**
   - Create C programs using sorting algorithms such as **merge sort**

3. Experimental Procedure

   i. Analyse the problem statement

ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
iii. Implement the algorithm in C language
iv. Compile the C program
v. Test the implemented program
vi. Document the Results
vii. Analyse and discuss the outcomes of your experiment

4. Questions
   1) Given a randomly ordered set of n numbers, design and develop an algorithm to sort them into non-descending order using quick sort and merge sort compare their efficiency. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of both the algorithms. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

   1. Limitations of Experiments

   2. Limitations of Results

3. Learning happened

4. Recommendations

## Laboratory 13

Title of the Laboratory Exercise: Heaps and Priority queues

1. Introduction and Purpose of Experiment

   A priority queue is different from a "normal" queue, because instead of being a "first-in-first-out" data structure, values come out in order by priority. A priority queue can be used in many applications where priority of the jobs should be considered for execution for example jobs sent to a printer based on the priority: Jobs sent by the department head should be printed first, then jobs sent by professors, then those sent by graduate students, and finally those sent by undergraduates. Heaps are useful data structures to implement a priority queue. In this experiment students learn how to implement priority queues using heaps and demonstrate the application of priority queue.

2. Aim and Objectives

   Aim

   - To design and develop programs for implementing a priority queue using heaps and demonstrate its operation

   Objectives

   At the end of this lab, the student will be able to

   - Implement priority queue using heaps
   - Demonstrate the pop operation of priority queue

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C/java language

iv. Compile the C/java program
v. Test the implemented program
vi. Document the Results
vii. Analyse and discuss the outcomes of your experiment

4. Questions
   1) Design and implement a priority queue using heap data structure. Demonstrate the operation of priority queues by setting the priority of input values and output the results. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of heaps in the implementation of priority queues. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.
5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

   1. Limitations of Experiments

   2. Limitations of Results

   3. Learning happened

   4. Recommendations

## Laboratory 14

Title of the Laboratory Exercise: Sets and Partitions

1. Introduction and Purpose of Experiment

   In Mathematics, a **set** is a collection of distinct objects. A partition of a set is a grouping of the set's elements into non-empty subsets, in such a way that every element is included in one and only one of the subsets. Set theory is seen as the foundation from which virtually all of mathematics can be derived. For example, structures in abstract algebra, such as groups, fields and rings, are sets closed under one or more operations. In this experiment students will learn to implement a disjoint set data structure and demonstrate its operation.

2. Aim and Objectives

   Aim

   - To design and implement a disjoint set data structure and demonstrate its operation.

   Objectives

   At the end of this lab, the student will be able to

   - Implement disjoint set
   - Demonstrate partitioning on set

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in C language
   iv. Compile the C program
   v. Test the implemented program
   vi. Document the Results

vii. Analyse and discuss the outcomes of your experiment

4. Questions

   Partition problem is to determine whether a given set can be partitioned into two subsets such that the sum of elements in both subsets is same. Design and implement an algorithm to solve the above stated set partition problem. Tabulate the output for various inputs and verify against expected values. Analyse the efficiency of the algorithm. Describe your learning along with the limitations of both, if any. Suggest how these can be overcome.

5. Calculations/Computations/Algorithms

6. Presentation of Results

7. Analysis and Discussions

8. Conclusions

9. Comments

   1. Limitations of Experiments

   2. Limitations of Results

   3. Learning happened

   4. Recommendations