



School of Computer Engineering
KIIT deemed to be University
Laboratory Lesson Plan – Autumn 2024 (3rd Semester)

Discipline: B.Tech (All branches)

Course name and Code: Data Structure Laboratory (CS29001)

L-T-P-Cr : 0-0-2-1

Instructor Name: Dr. Nikhilanand Arya, nikhilanand.aryafcs@kiit.ac.in , 7061112851

Instructor Chamber: Faculty Room no. 402, Block-C, Campus-14

Course Contents:

The course focuses on basic and essential topics in data structures and algorithms, including:

- Introduction (Structure, pointer, Dynamic Memory allocation and de-allocation)
- Arrays (following operations to be performed)
 - Insert, Delete, Linear search, Traversal, smallest and largest element, reverse the array element, sorting and searching an element in a dynamic array.
- Sparse Matrix (3-tuple format, transpose, addition operation)
- Single Linked list
 - Traversal of the list.
 - Check if the list is empty.
 - Insert a node at the certain position (at beginning/end/any position).
 - Delete a node at the certain position (at beginning/end/any position).
 - Delete the node for a given key.
 - Count the total number of nodes.
 - Search for an element in the linked list.
 - sort the list in ascending order
 - Reverse the list
 - Polynomial representation (single variable)
- Double and Circular Linked list
 - Insert a node at the certain position (at beginning/end/any position).
 - Delete a node at the certain position (at beginning/end/any position).
 - Traversal of the list.
- Stacks (using Array and single linked list)
 - Push
 - Pop
 - check stack is empty or not
 - check stack is full or not

- display stack elements
 - infix to postfix expression
- Linear Queues (using Array and single linked list)
 - Enqueue
 - Dequeue
 - IsEmpty
 - IsFull
 - Traverse
- Circular Queue (using array)
 - Enqueue
 - Dequeue
 - IsEmpty
 - IsFull
 - Traverse
- Deques (both Input restricted and output-restricted using static array)
 - Enqueue
 - Dequeue
 - IsEmpty
 - IsFull
- Priority Queue (using linked list)
 - Enqueue
 - Dequeue
 - Traverse
- Trees (Binary search tree using linked list)
 - preorder traversal
 - postorder traversal
 - inorder traversal
 - search an element
 - insert an element to the BST
 - display the largest element
 - display the smallest element
 - height of a node
 - count number of leaf nodes
- Graph (un-directed graph using Adjacency Matrix)
 - Display degree of each vertex
 - BFS traversal
 - DFS Traversal
- Sorting
 - Insertion sort
 - Selection sort
 - Merge sort
 - Quick sort
 - Heap sort
- Searching
 - Binary search

List of Experiments (Day wise):

❖ Introduction

Lab-1 Assignments

1.1 Write a program to read two numbers and compare the numbers using function call by address.

Sample Input:

Enter two numbers: 50 80

Sample Output:

50 is smaller than 80

Sample Input:

Enter two numbers: 40 10

Sample Output:

40 is greater than 10

Sample Input:

Enter two numbers: 50 50

Sample Output:

Both numbers are same

1.2 Write a program to create an array of n elements using dynamic memory allocation. Calculate sum of all the prime elements of the array using function and de-allocate the memory of the array after its use.

Sample Input:

Enter size of the array: 5

Enter array elements: 3 9 7 4 8

Sample Output:

Sum =10

1.3 Write a program to create a structure to store the information of n number of Employees. Employee's information includes data members: Emp-id, Name, Designation, basic_salary, hra%, da%. Display the information of employees with gross salary. Use array of structure.

Sample Input:

Enter no.of employees: 2

Enter employee 1 information:

Avneesh

Professor

10000

15%

45%

Enter employee 2 information:

Avantika

Professor

20000

10%

35%

Sample Output:

Employee Information:

Name: Suchismita

Designation: Professor

Basic Salary:10000

HRA %: 15%

DA %: 45%

Gross Salary: 14500

Name: Sarita

Designation: Professor

Basic Salary: 20000

HRA %: 10%

DA %: 35%

Gross Salary: 29000

1.4 Write a menu driven program to create a structure to represent complex number and perform the following operation using function :

1. addition of two complex number (call by value)
2. multiplication of two complex number (call by address)

Sample Input/Output:

Enter complex number 1: 3 4
Enter complex number 2: 4 5

MENU
1. addition
2. multiplication
Enter your choice: 1

Sum=7+9i
Enter your choice: 2

Sum=4+19i

❖ **Array**

Lab-2 Assignments

2.1 WAP to create a 1-D array of n elements and perform the following menu based operations using function.

- a. insert a given element at specific position.
- b. delete an element from a specific position of the array.
- c. linear search to search an element
- d. traversal of the array

Sample Input/Output:

Enter size n : 5
Enter element of array:
Enter Array elements: 10 23 45 37 52
MENU
1. Insert
2. Delete
3. Linear Search
4. Traverse
5. Exit
Enter option: 1
Element to insert: 61 Enter Position: 2
Element inserted
Enter option: 4
Array Elements: 10 23 61 45 37 52

Note: Other menu choices are similarly needs to verify.

2.2 Write a program to perform the following operations on a given square matrix using functions:

- i. Find the no.of nonzero elements

- ii. Display upper triangular matrix
- iii. Display the elements of just above and below the main diagonal

Sample Input:

Enter size of the square matrix: 4

Enter elements of the matrix:

```
8 2 1 0
1 0 7 6
0 6 2 4
3 9 5 0
```

Sample Output:

Nonzero elements : 12

Upper triangular matrix:

```
2 1 0
  7 6
    4
```

2.3 WAP to represent a given sparse matrix in 3-tuple format using 2-D array.

Sample Input:

Enter size of the sparse matrix: 4 5

Enter elements of sparse matrix: 0 0 33 0 0 0 17 0 0 0 0 0 0 46 0 0 0 0 0 51

Sample Output:

sparse matrix in 3-tuple format

```
4 5 4
0 2 33
1 1 17
2 3 46
3 4 51
```

Lab-3 Assignments

3.1 WAP to perform transpose of a given sparse matrix in 3-tuple format.

Sample Input:

Enter sparse matrix in 3-tuple format

```
4 5 4
0 2 33
1 1 17
2 3 46
3 4 51
```

Sample Output:

Transpose of sparse matrix:

```
R  C  Element
5  4  4
1  1  17
2  0  33
3  2  46
4  3  51
```

3.2 WAP to perform addition of two given sparse matrix in 3-tuple format.

Sample Input:

Enter sparse matrix-1 in 3-tuple format

```
4 5 4
0 3 30
1 1 10
2 3 40
3 4 21
```

Enter sparse matrix-2 in 3-tuple format

```
4 5 5
0 2 65
1 1 12
2 3 45
3 3 71
```

Sample Output:

Resultant Matrix in 3-tuple format

```
4 5 5
0 2 65
0 3 30
1 1 22
2 3 85
3 3 71
3 4 21
```

3.3 WAP to represent the polynomial of single variable using 1-D array and perform the addition of two polynomial equations.

Sample Input:

Enter maximum degree of x: 2

Enter Polynomial-1 from lowest degree to highest degree : 4 2 3 (Hint: $4+2x+3x^2$)

Enter Polynomial-2: 6 5 2

Sample Output:

Resultant Polynomial: $5x^2+7x^1+10x^0$

❖ **Linked list**

Lab-4 Assignments

4.1 Write a program to create a single linked list of n nodes and perform the following menu-based operations on it using function:

- i. Insert a node at specific position
- ii. Deletion of an element from specific position
- iii. Count nodes
- iv. Traverse the linked list

Sample Input/Output:

Enter number of nodes: 5
Enter the elements: 17 23 47 11 78 92 51
MENU:
1. Insert the node at a position
2. Delete a node from specific position
3. Count
4. Traversal
5. Exit
Enter choice: 1
Enter element: 66
Enter position: 2
Node inserted
Enter choice: 4
The list is: 17-> 66->23-> 47-> 11-> 78-> 92-> 51
Enter the choice: 3
The total number of nodes: 8

Note: Other menu choices are similarly needs to verify.

4.2 In addition to **4.1**, perform following operations using function on the single linked list:
i. search an element in the list
ii. sort the list in ascending order
iii. reverse the list

Sample Input/Output:

Enter number of nodes: 5
Enter the elements: 17 23 47 11 78 92 51
MENU:
1. Insert the node at a position
2. Delete a node from specific position
3. Count
4. Traverse
5. Search
6. Sort
7. Reverse
8. Exit
Enter choice: 1
Enter element: 66
Enter position: 2
Node inserted
Enter choice: 4
The list is: 17-> 66->23-> 47-> 11-> 78-> 92-> 51
Enter the choice: 5
Enter element to be searched: 23
Element found at node-3
Enter the choice: 7
Reverse list: 51->92->78->11->47->23->66->17

Note: Other menu choices are similarly needs to verify.

4.3 Write a program to represent the polynomial equation of single variable using single linked list and perform the addition of two polynomial equations.

Sample Input:

Polynomial-1:

Enter the Maximum power of x: 2

Enter the coefficient of degree 2: 4

Enter the coefficient of degree 1: 3

Enter the coefficient of degree 0:2

Polynomial-2:

Enter the Maximum power of x: 3

Enter the coefficient of degree 3: 5

Enter the coefficient of degree 2: 4

Enter the coefficient of degree 1:6

Enter the coefficient of degree 0:10

Sample Output:

Sum: $5x^3+8x^2+9x^1+12x^0$.

Lab-5 Assignments

5.1 Write a program to create a double linked list and perform the following menu-based operations on it:

- i. insert an element at specific position
- ii. delete an element from specific position
- iii. Traverse the list

Sample Input/Output:

Enter number of nodes: 5

Enter the elements: 17 23 47 11 78 92 51

MENU:

1. Insert the node at a position
2. Delete a node from specific position
3. Traversal
5. Exit

Enter choice: 1

Enter element: 66

Enter position: 2

Node inserted

Enter choice: 3

The list is: 17-> 66->23-> 47-> 11-> 78-> 92-> 51

Note: Other menu choices are similarly needs to verify.

5.2 Write a program to create a circular linked list and display the elements of the list.

Sample Input:

Enter no.of nodes: 5

Enter info of node1: 30

Enter info of node1: 50

Enter info of node1: 40

Enter info of node1: 20

Enter info of node1: 70

Sample Output:

Clinkedlist: 30 50 40 20 70

5.3 Write a program to represent the given sparse matrix using header single linked list and display it.

Sample Input:

Enter size of the sparse matrix: 4 5

Enter elements of sparse matrix: 0 0 33 0 0 0 17 0 0 0 0 0 0 46 0 0 0 0 0 51

Sample Output:

sparse matrix in 3-tuple format

```
4 5 4
0 2 33
1 1 17
2 3 46
3 4 51
```

❖ Stack**Lab-6 Assignments**

6.1 Write a menu driven program to create a stack using array and perform the following operation using function

- Push
- Pop
- check stack is empty or not
- check stack is full or not
- display stack elements

Sample Input/Output:

Main Menu

1. Push
2. Pop
3. IsEmpty
4. IsFull
5. Traverse
6. Exit

Enter option: 1

Enter element to be pushed into the stack: 22

Enter option: 1

Enter element to be pushed into the stack: 33

Enter option: 1

Enter element to be pushed into the stack: 44

Enter option: 1

Enter element to be pushed into the stack: 88

Enter option: 1

Enter element to be pushed into the stack: 66
Enter option: 5
Stack: 66 88 44 33 22
Enter option: 2
66 deleted from Stack
Enter option: 3
Stack empty: False

Note: Other menu choices are similarly needs to verify.

6.2 Write a menu driven program to create a stack using linked list and perform the following operation using function

- a. Push
- b. Pop
- c. IsEmpty
- d. display the stack elements

Sample Input/Output:

Main Menu

- 1. Push
- 2. Pop
- 3. IsEmpty
- 4. Traverse
- 5. Exit

Enter option: 1
Enter element to be pushed into the stack: 12
Enter option: 1
Enter element to be pushed into the stack: 35
Enter option: 1
Enter element to be pushed into the stack: 24
Enter option: 1
Enter element to be pushed into the stack: 8
Enter option: 1
Enter element to be pushed into the stack: 41
Enter option: 4
Stack: 41 8 24 35 12
Enter option: 2
41 deleted from Stack

Note: Other menu choices are similarly needs to verify.

6.3 Write a program to convert infix expression to postfix expression using stack.

Sample Input:

Enter infix expression: (a+b)/c*d-e^f

Sample Output:

Postfix: $ab+c/d*ef^-$

❖ Queue**Lab-7 Assignments**

7.1 Write a menu driven program to create a linear queue using array and perform Enqueue, Dequeue, Traverse, IsEmpty, IsFull operations.

Sample Input/Output:

Enter the size of Queue: 5
Main Menu
1. Enqueue
2. Dequeue
3. IsEmpty
4. IsFull
5. Traverse
6. Exit

Enter option: 1
Enter element: 15
Enter option: 1
Enter element: 23
Enter option: 1
Enter element: 40
Enter option: 5
Queue: 15 23 40
Enter option: 2
Element deleted
Enter option: 5
Queue: 23 40

Note: Other menu choices are similarly needs to verify.

7.2 Write a menu driven program to implement linear queue operations such as Enqueue, Dequeue, IsEmpty, Traverse using single linked list.

Sample Input/Output:

Main Menu
1. Enqueue
2. Dequeue
3. IsEmpty
4. Traverse
5. Exit

Enter option: 1

Enter element: 55
Enter option: 1
Enter element: 23
Enter option: 1
Enter element: 46
Enter option: 4
Queue: 55 23 46
Enter option: 2
Element deleted
Enter option: 4
Queue: 23 46

Note: Other menu choices are similarly needs to verify.

7.3 Write a menu driven program to implement circular queue operations such as Enqueue, Dequeue, Traverse, IsEmpty, IsFull using array.

Sample Input/Output:

Enter Queue size: 3

Main Menu

1. Enqueue
2. Dequeue
3. IsEmpty
4. IsFull
5. Traverse
6. Exit

Enter option: 1
Enter element: 25
Enter option: 1
Enter element: 36
Enter option: 1
Enter element: 42
Enter option: 5
CQueue: 25 36 42
Enter option: 2
Element deleted
Enter option: 5
CQueue: 36 42
Enter option: 2
Element deleted
Enter option: 5
CQueue: 42
Enter option: 3
Queue Empty: False

Lab-8 Assignments

8.1 Write a menu driven program to implement Deques (both Inputrestricted and output-restricted) and performed operations such as Enqueue, Dequeue, Peek, IsEmpty, IsFull using static array.

Sample Input/Output:

Input restricted Dequeue Menu

1. Insert at right
2. Delete from left
3. Delete from right
4. Display
- 5.Quit

Enter choice: 1
Enter element:87
Enter choice: 1
Enter element:32
Enter choice: 4
Deque: 87 32
Enter choice: 2
87 deleted
Enter choice: 4
Deque: 32

Note: Other menu choices are similarly needs to verify.

8.2 Write a menu driven program to implement priority queue operations such as Enqueue, Dequeue, Traverse using linked list.

Sample Input/Output:

Main Menu

1. Enqueue
2. Dequeue
3. Display
4. Exit

Enter option: 1
Enter element: 34
Enter priority: 1

Enter option: 1
Enter element: 23
Enter priority: 3
Enter option: 1
Enter element: 46
Enter priority: 2
Enter option: 3
Priority Queue:
Priority Item

| | |
|---|----|
| 1 | 34 |
| 2 | 46 |
| 3 | 23 |

Note: Other menu choices are similarly needs to verify.

❖ **Tree**

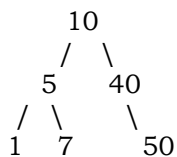
Lab-9 Assignments

9.1 Write a program to create a binary search tree of n data elements using linked list and perform the following menu driven operations:

- preorder traversal
- postorder traversal
- inorder traversal
- search an element

Sample Input/Output:

Enter number of nodes: 6
Enter elements of BST: 10 5 1 7 40 50
BST Created:



MAIN MENU

1. Preorder
2. Postorder
3. Inorder
4. Search
5. Exit

Enter option: 1

Preorder: 10 5 1 7 40 50

Enter option: 2

Postorder: 1 7 5 50 40 10

Note: Other menu choices are similarly needs to verify.

9.2 In addition to the **9.1**, perform the following menu driven operations on BST.

- insert an element to the BST
- display the largest element
- display the smallest element
- height of a node
- count number of leaf nodes

Sample Input/Output:

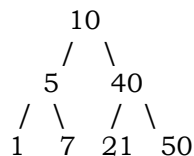
MAIN MENU

1. Insert
2. Largest

3. Smallest
4. Height
5. Count leaf nodes
6. Exit

Enter option: 1

Enter element to insert in BST: 21



Enter Option: 2

Largest element in BST=50

Note: Other menu choices are similarly needs to verify.

9.3 In addition to **9.2**, perform deletion of an element in the BST using function.

❖ Graph

Lab-10 Assignments:

10.1 WAP to create an un-directed graph using Adjacency Matrix Method and display the degree of each vertex.

Sample Input:

Enter number of vertex: 5

Vertices 1 & 2 are Adjacent ? (Y/N) :y

Vertices 1 & 3 are Adjacent ? (Y/N) :n

Vertices 2 & 1 are Adjacent ? (Y/N) :y

Vertices 2 & 3 are Adjacent ? (Y/N) :y

Vertices 3 & 1 are Adjacent ? (Y/N) :n

Vertices 3 & 2 are Adjacent ? (Y/N) :y

Sample Output:

| Vertex | Degree |
|--------|--------|
| 1 | 1 |
| 2 | 2 |

3 1

10.2 In addition to **10.1**, display DFS traversal sequence of the undirected graph.

Sample Input:

Adjacency Matrix:

```
0 1 1 1 0
1 0 0 0 1
1 0 0 0 1
1 0 0 0 1
0 1 1 1 0
```

Enter start vertex: 0

Sample Output:

Deapth First Search: 0 1 4 2 3

10.3 In addition to **10.1**, display BFS traversal sequence of the undirected graph.

Sample Input:

Enter number of vertex: 5

Enter Adjacency Matrix:

```
0 0 1 1 0
0 0 0 1 1
1 0 0 1 0
1 1 1 0 0
0 1 0 0 0
```

Enter start vertex: 0

Sample Output:

Breadth First Search: 0 2 3 1 4

10.4 WAP to create a directed graph using Adjacency Matrix Method and display the degree of each vertex.

Sample Input:

Enter number of vertex: 3

Vertices 1 & 2 are Adjacent ? (Y/N) :n

Vertices 1 & 3 are Adjacent ? (Y/N) :y

Vertices 2 & 1 are Adjacent ? (Y/N) :y

Vertices 2 & 3 are Adjacent ? (Y/N) :y

Vertices 3 & 1 are Adjacent ? (Y/N) :n

Vertices 3 & 2 are Adjacent ? (Y/N) :y

Sample Output:

| Vertex | In_Degree | Out_Degree | Total_Degree |
|--------|-----------|------------|--------------|
| 1 | 1 | 1 | 2 |
| 2 | 1 | 2 | 3 |
| 3 | 2 | 1 | 3 |

❖ Sorting**Lab-11 Assignments:**

11.1 Write a program to sort array of elements in ascending and descending order by insertion sort using function.

Sample Input:

Enter no.of elements: 5
Enter elements: 22 55 33 88 44

Sample Output:

Ascending order: 22 33 44 55 88
Descending order: 88 55 44 33 22

11.2 Write a program to sort array of elements in ascending and descending order by selection sort using function.

Sample Input:

Enter no.of elements: 5
Enter elements: 11 55 22 66 33

Sample Output:

Ascending order: 11 55 22 33 66
Descending order: 66 55 33 22 11

11.3 Write a program to perform quick sort on array of elements to arrange it in ascending order using function.

Sample Input:

Enter no.of elements: 5
Enter elements: 70 50 40 20 10

Sample Output:

Ascending order: 10 20 40 50 70

Lab-12 Assignments:

12.1 Write a program to perform merge sort on array of elements to arrange it in ascending order using function.

Sample Input:

Enter no.of elements: 6
Enter elements: 10 30 40 60 20 80

Sample Output:

Ascending order: 10 20 30 40 60 80

12.2 Write a program to perform heapsort on array of elements to arrange it in ascending order using function.

Sample Input:

Enter no.of elements: 8
Enter elements: 52 31 25 14 27 38 46 50

Sample Output:

Ascending: 14 25 27 31 38 46 80

12.3 Write a program to search a given element within array using binary search.

Sample Input:

Enter elements: 52 31 25 14 27 38 46 50
Enter element to be searched: 27

Sample Output:

Element found

Grading Policies:

| Continuous Evaluation components | | | | |
|---|-------------------------|-------------|----------|--------------|
| Sr# | Area | Mark | # | Total |
| 1 | Internal Sending | | | |
| 1.1 | Lab record evaluation | 1 | 10 | 10 |
| 1.2 | Quiz | 5 | 2 | 10 |
| 1.3 | Viva | 5 | 2 | 10 |
| 1.4 | Program Execution | 2 | 10 | 20 |
| 1.5 | Class Participation | 1 | 10 | 10 |
| Total | | | | 60 |
| End semester evaluation | | | | |
| 2.1 | Program Execution | 15 | 1 | 15 |
| 2.2 | Viva | 10 | 1 | 10 |
| 2.3 | Quiz | 15 | 1 | 15 |
| Total | | | | 40 |

Reference Materials:-**Reference Book:**

- RB1. Data Structures, Schaum's OutLines, Seymour Lipschutz, TATA McGraw Hill
- RB2. Data Structures using C by Aaron M. Tenenbaum, Yedidyah Langsam, Moshe J. Augenstein. Pearson, 1st Edition
- RB3. Data Structures A Pseudocode Approach with C, 2nd Edition, Richard F. Gilberg, Behrouz Forouzan, CENGAGE Learning, India Edition
- RB4. Data Structures Using C, Second Edition, Reema Thereja, Oxford University Press
- RB5. Data Structures and Algorithm Analysis in C, Mark Allen Weiss, Pearson Education, 2nd Edition.

Reference Site:

- RS1. NPTEL - <https://onlinecourses.nptel.ac.in/explorer>
- RS2. Tutorials Point - https://www.tutorialspoint.com/data_structures_algorithms/
- RS3. Geeks for geeks - <http://www.geeksforgeeks.org/>

Dr. Nikhilanand Arya
(Course Faculty)