Dharmsinh Desai University, Nadiad
Faculty of Technology
Department of Computer Engineering
(2014-15)
Last updated - 22/08/2014

Subject Name: DSA (Data Structures and Algorithms) Semester: B.Tech-III

Hardware/Software required: LINUX OS.
GCC Compiler.

Lab 1:

Core: 1) Perform insert_front, delete_front, insert_end, delete_end operations on a linked list.
2) Implement print() function to print linked list. Also implement rprint() function to print linked list in reverse order using recursion. (Also known as traverse)

Plus: 1) Insert node at n^{th} position & Delete n^{th} node from the linked list.
2) Represent polynomial equations as a linked list. Add two polynomial equation and store result in 3^{rd} one. (ex $(3x^2+2x+4) + (5x^2+7x+10) = 8x^2+9x+14$

Lab 2:

Core: 1) Implement Stack and Queue using Array.

Plus: 1) Implement Stack and Queue using linked list.
2) Analyze how does the backward and forward button works in internet browsers/clients like chrome, firefox, etc. Knowing the requirement, develop an algorithm, data structure(s) and of course C program to simulate the same. i.e. Let use press B for backward, F for forward, any other character to mimic mouse click/event request. Show the current data character on the screen. It can be from backward or forward data structure or the recent pressed key.

Lab 3:

Core: 1) Implement circular linked list and doubly linked list.

Plus: 1) Delete all node which having data value "x"
2) Delete first node which having data value "x":
3) Delete last node which having data value "x"
4) Count number of node which having data value "x" and from that delete 2^{nd} node.
5) Implement process scheduling algorithm using appropriate data structure.
"Consider that there are n processes present at a particular time and CPU allows 1 sec execution time to process, before switch to next one. Take required execution time for all the processes from the user. The CPU needs to give a fair chance of execution to each. As an output print whole execution scenario with time slot".

Lab 4:

Core:

1. Write a program to evaluate postfix expression.
2. Implement solution to classic Tower of Hanoi problem using recursion.

Plus:

1. Write a program to validate mathematical expression for different types of brackets' ordering.
2. Write a program to evaluate prefix expression.
3. Write a program to convert infix notation into postfix notation.

Lab 5:

Core:

1. Write a program to create a binary search tree and display sorted data (Both ascending and descending separately).

Plus:

1. Write a program to create expression tree. Traverse in all three orders. (in-pre-post).
2. Write a program to merge two given binary search trees.

Lab 6:

Core:

1. Write a program to create adjacency list and adjacency matrix.
2. Write a program to create a graph and perform DFS and BFS traversal.
3. Write a program to convert given adjacency matrix into adjacency list.

Plus:

1. Write a program to implement single source shortest path using Dijkastra's algorithm.
2. Write a program to identify Articulation Points (or Cut Vertices) in a given graph.

"A vertex in an undirected connected graph is an articulation point (or cut vertex) iff removing it (and edges through it) disconnects the graph. Articulation points represent vulnerabilities in a connected network – single points whose failure would split the network into 2 or more disconnected components. They are useful for designing reliable networks. For a disconnected undirected graph, an articulation point is a vertex removing which increases number of connected components."

Lab 7:

Core:

1. Write a program to implement min and max heap data structure.
2. Implement a forest consisting of different types of trees.

Plus:

1. Write a program to implement Huffman encoding and decoding techniques.
2. Write a function to check whether a given binary tree is balanced or not? Write another function to balance a given tree if it is not already balanced. Test using driver program.
3. Implement graph using adjacent multi list. For every vertex, print its adjacent vertices.

Lab 8:

Core:

1. Implement following sorting techniques:
    - Bubble sort and improved bubble sort.
    - Quick sort
    - Merge sort
    - Insertion sort

Plus:

1. Implement following sorting techniques:
    - Heap sort
    - Shell sort

Lab 9:

Core:

1. Implement static hashing method.

Plus:

1. Implement dictionary using three level hash function.

Level1: For character position 1 -> A,B,C,...,Z (26)
Level2: For character position 2 -> AtoE,FtoJ,KtoO,PtoT,UtoZ (avg. 5 difference)
Level3: For character position 3 -> AtoH, ItoP, QtoZ. (Avg 8 difference)
Remaining store sequentially.
Please, note that words/symbols of length 3 or less, they get positioned in the bucket at appropriate levels immediately. i.e. 'an' ,'is', 'the'.

Lab 10:

Core:

1. Implement following search techniques:
    - Sequential search
    - Binary Search

Plus:

1. Implement Red-Black tree.
2. Implement AVL tree.
3. Implement indexed based searching.