Assignment List: Data Structures

Day 1

1. Practice GDB tool for all programs.
2. Write a program to swap two pointers.
   - Write function 'swap' which takes two arguments as pointers and doesn't return any thing. Swap the contents of these two pointers in 'swap' function.
   - Print the contents of two pointers in main before & after call of 'swap' function.
3. Write a program to find max of three numbers
   - Write a function “compare()” which takes 3 addresses as arguments and returns address of max element.
   - Call and print the result in “main” function.
4. Write a program to demonstrate dynamic memory allocation.
   - InitArray function that creates integer array of n elements in heap memory. It returns this array to 'main'.
   - FillElement() function that fills this array from User Input.
   - DisplayArray() function that prints this array to screen.
   - ReverseArray() function that reverses the array.
   - DeleteArray() function that free the array memory.
5. Write a function print_reverse(string) to print the string in reverse order using recursion.
6. Implement Rational Number ADT: Write functions to Read rational number from user, check rational number, write add function, multiply function, divide functions

Day 2

Implement using menu for each program and ask for options from user

1. Write a program to implement following functions for Linked lists of integers.
   - Write function 'insert_end' to insert an element at end of linked list. Take Linked List pointer and integer as two arguments to function.
   - Write function 'insert_beg' to insert at beginning of linked list. Take Linked List pointer and integer as two arguments to function.
   - Write function 'delete_beg' & 'delete_end' to delete elements from

    linked list. Take Linked List pointer only as single argument.
    - Write function 'search' to search as element in Linked list.
    - Write function 'display' to print the elements of linked list. Take Linked List pointer only as single argument.
2. Write a function 'print_reverse' for printing the elements of a linked list in the REVERSE ORDER. Do not modify the linked list.
3. Write a function 'reverse' to REVERSE a Linked List.

Day 3

1. Write a program to add two polynomials. Use linked list representation for polynomials.
2. Implement a doubly linked list ADT with functions for creation, insertion at end, deletion from end & searching of nodes.
3. Develop an Linked list implementation of self-adjusting lists. A self-adjusting list is like a regular list, except that all insertions are performed at the front, and when an element is accessed by the Find, it is moved to the front of the list without changing the relative order of the other items.
4. Write an Implementation of Circular Singly Linked List. Implement insert_front, insert_end and display functionalities.

Day 4

1. Create a Stack ADT using array and implement push, pop and tos operations.
2. Write a program to Implement Evaluation of postfix expression.
3. Write a program for Balancing parentheses in an Arithmetic Expression. Use parenthesis: [,], {, }, (,) .
4. Write a program for Converting Infix to postfix.

Day 5

1. Implement a circular Queue ADT using array.
2. Find out whether a string is PALINDROME or not using stack & Queues.
    - Write a function “is_palindrome” which takes a string as argument and returns 0 if True, 1 if False.
3. Write a program to implement Stack using Linked List.
4. Write a program to implement Queue ADT using Linked List.

Day 6

1.Write a function “bubble_sort” to implement Bubble sort. Pass array “arr” and size “n” as arguments from main.

2.Write a function “insertion_sort” to implement Insertion Sort. Pass array “arr” and size “n” as arguments from main.

3.Write a function “selection_sort” to implement Selection Sort. Pass array “arr” and size “n” as arguments from main.

4.Write a function “merge_sort” to implement Merge sort. Pass array “arr” and size “n” as arguments from main.

5.Write a function “quick_sort” to implement Quick Sort. Pass array “arr” and size “n” as arguments from main.

Day 7

1. Write C program to sort 'n' numbers using Heap Sort.
2. Implement Searching an element in an array using linear search.
3. Implement Searching an element in an array using binary search.
4. Implement Hashing technique using Linear Probing.
5. Implement Hashing technique using Separate Chaining.

Day 8

Given a postfix expression, construct an expression tree and include the following functions

1. inorder_traversal
2. preorder_traversal
3. postorder_traversal
4. height_of_tree
5. mirror_image
6. number_of_ nodes

Day 9

1. Create Binary Search Tree(BST) with the following functions
   a) insert_into_bst
   b) deletion_from_bst
   c) find_in_bst
   d) find_min_in_bst

e) find_max_in_bst

2. Write a non recursive routine to insert an element in binary search tree.
3. Create an AVL tree by including functionality for inserting, singlerotationleft, singlerotationright, doublerotationleft & doublerotationright.

Day 10

1. Develop a program to find the shortest path in a graph.

Micro Project: Implement file system environment similar to linux command line. Add commands like *ls, cd, mkdir, pwd, clear, tree* etc. Add the facility of putting files & directories.