Assignments List: Data Structures

Note:

(a) Use GDB tool to debug all programs.

(b) Avoid use of GLOBAL VARIABLES.

(c) Do Not copy assignments from other sources. Violations may lead to deduction in marks.

Day 1

1. Write a program to find largest and smallest of n numbers without sorting.
2. Write a program to merge two sorted arrays: A of size n, B of size m. Merge arrays A and B to third array C of size n+m.
3. Write a function PrintReverse(string) to print the string in reverse order using recursion.
4. Write a program to swap two pointers.
   - Write a function 'swap' which takes two arguments as integer 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.

Day 2

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

1. Write a program to implement Singly 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.
   - Write function *insert_beg()* to insert at beginning of linked list. Take Linked List pointer and integer as two arguments.
   - Write function *delete_beg()* & *delete_end()* to delete element from linked list. Take Linked List pointer only as single argument.
   - Write function *search()* to search as element in Linked list and return node address. Return NULL for failed search.
   - Write function *insert_after()* to insert an element after above *search()* function. Insert element at end for failed *search()*.
   - 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.
3. Write a function *reverse()* to reverse a Linked List.
4. Swap two adjacent elements by adjusting only the pointers(and not the data).(Hint: Create any linked list of n+3 nodes where n ≥ 0. swap 2nd and 3rd nodes)
5. Given two sorted linked lists L1 and L2 by data, perform L1 ∩ L2. Write a function intersect(L1, L2). Store the result of operation in L3. (Hint: For simplicity, take unique data elements in both lists).

Day 3

1. Use Singly linked list representation for polynomials to create P1 and P2. Write a program to add two polynomials: P3 = add_poly(P1,P2).
2. Implement a doubly linked list program with functions: *insert_at_beg, insert_at_end, delete_at_beg, delete_at_end & search*.
3. Develop a Singly 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. Implement above functionality as *self_adjusting_find()* for singly linked lists.
4. Write an Implementation of Circular Linked List. Implement *insert_front*, *insert_end* and *display* functions.

Day 4

1. Create a Stack using array implementation with functions: *push()*, *pop* and *tos.* Write a function *display_stack* to print all elements.
2. Write a program to Implement Evaluation of postfix expression.
3. Balancing parentheses in an Arithmetic Expression.
4. Write a function *is_palindrome* which takes a string as argument and returns 1 if True,0 if False.
5. Implement a Circular Queue using array implementation with functions: enqueue, dequeue. Write a function *display_queue* to print all elements.

Day 5

1. Write a program to implement Stack using Linked List .
2. Write a program to implement Queue using Linked List.
3. Implement Searching an element from the list using linear search.
4. Implement Searching an element in a list using binary search.

5. Implement Hashing technique using Open addressing and Separate Chaining.

Day 6

Implement C program to sort array of 'n' numbers using

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.

Day 7

1. Write a function “quick_sort” to implement Quick Sort. Pass array “arr” and size “n” as arguments from main.
2. Write C program to sort 'n' numbers using Heap Sort.

Day 8

1. Create Binary Search Tree(BST) with the following functions
    1. insert_into_bst
    2. deletion_from_bst
    3. find_in_bst
    4. find_min_in_bst
    5. find_max_in_bst
2. Write a non recursive routine to insert an element in binary search tree.

Day 9

1. Add following functions to Binary Search Tree(BST)
    - height_of_tree
    - mirror_image
    - number_of_ nodes
    - inorder_traversal
    - preorder_traversal
    - postorder_traversal
2. Create an AVL tree by including functionality for

inserting, singlerotationleft, singlerotationright, doublerotationleft & doublerotationright.

Nano 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.*