Questions on Linked Lists

1. Implement **Singly Linked List** as an ADT. A file "singly_list.h" has already been uploaded on BB for your reference. You need to write 2 other files – **"singly_list.c"** (define all the functions whose prototypes are already given in "singly_list.h" in this file) and "**singly_test_list.c"** (write main function in this file where you will call all the functions defined in "singly_list.c").

2. Implement **Doubly Linked List** as an ADT. A file "doubly_list.h" has already been uploaded on BB for your reference. You need to write 2 other files – **"doubly_list.c"** (define all the functions whose prototypes are already given in "doubly_list.h" in this file) and "**doubly_test_list.c"** (write main function in this file where you will call all the functions defined in "doubly_list.c").

3. Implement **Circular Linked List** as an ADT. A file "circular_list.h" has already been uploaded on BB for your reference. You need to write 2 other files – **"circular_list.c"** (define all the functions whose prototypes are already given in "circular_list.h" in this file) and "**circular_test_list.c"** (write main function in this file where you will call all the functions defined in "circular_list.c").

4. Create 3 files (sorted.h, sorted.c, test_sorted.c) to implement a Sorted (in Increasing Order) Singly Linked List ADT (Implement only Insertion and Display Functions).

5. Write a C program that will remove all duplicate elements present in a singly linked list. You can use the below given insert() function in your program to test your code.

```
typedef struct node
{
        int data;
        struct node *next;
}node;

node *head=NULL;

void insert(int element)
{
        node *new = (node *)malloc(sizeof(node));
        new->data = element;
        new->next = NULL;
        if(head == NULL)
        head=new;
        else
        {
                node *temp=head;
                while(temp->next != NULL)
                temp = temp->next;
                temp->next = new;
        }   }
```

6. Create 3 different files (list.h, list.c, test_list.c) to implement List ADT. Define 3 functions in it – insert(), swap() and display() as per below given specifications:

   i. Insert( ) function must take the element from the user and insert it in the linked list one after another (just think of "insert at the end" of the linked list).
   ii. Swap( ) function must take 2 values (say X and Y) from the user and must swap the 2 respective nodes containing data X and Y. Make sure the 2 nodes must be swapped and not just the data of the node is swapped.
   Your function must ensure the different positions of X and Y, specifically the following:
   (i) X and Y may or may not be adjacent.
   (ii) Either X or Y may be a head node.
   (iii) Either X or Y may be the last node.
   (iv) X and/or Y may not be present in the linked list.

   For Example :
   Input: 20->12->40->90->56 (Values to be swapped are 20 and 40)
   Output: 40->12->20->90->56

   iii. Display( ) function must display the current status of the linked list.

7. Write a C Program to remove all the duplicate elements from sorted linked list.

8. Take some elements from the user and store them in the form of singly linked list. In the same manner, take some more elements from the user and store them in another singly linked list so that you will have 2 separate singly linked lists. For this, make use of a function create1() which will keep on adding the elements given by the user one after another in 1st list and create2() which will keep on adding the elements given by the user one after another in 2nd list. Once the lists are being created, merge the elements of second list at the alternate positions in the first list. That means insert nodes of second list into first list at alternate positions of first list.

   For Example
   First List: 1->5->7->3->9
   Second List: 6->10->2->4,
   After Merging, the first list should become 1->6->5->10->7->2->3->4->9 and second list should become empty.

   The nodes of second list should only be inserted when there are positions available.

   For Example:
   First List: 1->5->7->3
   Second List: 6->10->2->4,
   After Merging, the first list should become 1->6->5->10->7->2->3 and second list should become 4.