Referecnce Code for Data Structres

```c
#include<stdio.h>
#include<stdlib.h>

struct Node     // Structre Defination
{
   int data;
   struct Node * next;
};

typedef struct Node NODE;
typedef struct Node * PNODE;
typedef struct Node ** PPNODE;

/*-----------------------------------------------------------
         Old Name                       New Name
 -----------------------------------------------------------
         struct Node                      NODE
         struct Node *                    PNODE
         struct Node **                   PPNODE
 -----------------------------------------------------------*/

/////////////////////////////////////////////////////////////////
//
// Function name  : InsertFirst
// Description       : Used to insert at first position of Linked List
// Parameters        : Addreass of First pointer & data of node
// Return Value      : void
//
/////////////////////////////////////////////////////////////////

void InsertFirst(PPNODE Head, int no)
{
   PNODE newn = NULL;
   newn = (PNODE)malloc(sizeof(NODE)); // Allocate memory
   newn-> data = no;   // Iniitialise data
   newn-> next = NULL; // Initialise pointer
```

```
    if(*Head == NULL) // Linkedlist is empty
    {
        *Head = newn;
    }
    else  // LL contains atleast one node
    {
        newn -> next = *Head;
        *Head = newn;
    }
}

////////////////////////////////////////////////////////////////////
//
// Function name   : InsertLast
// Description     : Used to insert at last position of Linked List
// Parameters      : Addreass of First pointer & data of node
// Return Value    : void
//
////////////////////////////////////////////////////////////////////

void InsertLast(PPNODE Head, int no)
{
    PNODE newn = NULL;
    PNODE temp = *Head;

    newn = (PNODE)malloc(sizeof(NODE)); // Allocate memory
    newn-> data = no;   // Iniitialise data
    newn-> next = NULL; // Initialise pointer

    if(*Head == NULL) // Linkedlist is empty
    {
        *Head = newn;
    }
    else  // LL contains atleast one node
    {
            while(temp->next != NULL)
            {
                temp = temp->next;
        }
        temp -> next = newn;
    }
}

/////////////////////////////////////////////////////////////////////
//
// Function name  : Display
// Description    : Used to idisplay elements of Linked List
// Parameters     : First pointer
// Return Value   : void
//
/////////////////////////////////////////////////////////////////////

void Display(PNODE Head)
{
    while(Head != NULL)
    {
        printf("%d\t",Head->data);
        Head = Head -> next;
    }
}

/////////////////////////////////////////////////////////////////////
//
// Function name  : Count
// Description    : Used to count elements of Linked List
// Parameters     : First pointer
// Return Value   : int
//
/////////////////////////////////////////////////////////////////////

int Count(PNODE Head)
{
    int iCnt = 0;

    while(Head != NULL)
    {
        iCnt++;
        Head = Head -> next;
    }

    return iCnt;
}

int main()
{
    int iRet = 0;

    PNODE First = NULL;
    InsertFirst(&First, 51);
    InsertFirst(&First, 21);
    InsertFirst(&First, 11);

    Display(First);

    iRet = Count(First);

    printf("\nNumber of elements are %d : \n",iRet);

    InsertLast(&First,101);
    InsertLast(&First,111);

    Display(First);
    return 0;
}
```

Rules for data structures

1. All memory allocations should be dynamic (malloc)

2. Inside main function we have to maintain one pointer throughout the application ie First.

3. memory for the node should be allocated and deallocated inside the helper functions

4. If the function is going to mody the linkedlist then we have to pass the address of First pointer. (InsertFirst, InsertLast, DeleteFirst, DeleteLast, InsertAtPos,DeleteAtPos).

5. If the function is not going to modify the linkedlist then pass the First pointer directlt (Display , Count).

6. If our function accepts address of First pointer then for traversal purpose use the temporary pointer otherwise it afftects the value of Head pointer from the main function.