Logic Building

Assignment Part 3

Problems on Linked List
(Data structures using C)

While writing the program follow some instructions as:-

- First write algorithm for given problem statement.
- Write appropriate function name using camel case. Ex. MaxTwoNumber( )
- Write proper name for variables. Ex. int iNumber= 0; float fValue= 0.0;
- Use proper indentations.
- Use proper comments for important statements.
- Remove all warnings after compilation.
- Reuse the variables if possible.
- Write header for every function which contains
    - Function name
    - Input parameters
    - Output value
    - Description of function
- After writing the program write input and expected output.
- Initialize all unused pointer variables to NULL to avoid run time accidents.
- In data structures programs use call by address wherever required.
- Use proper loop checking conditions to avoid segmentation faults.
- Free nodes physically to avoid memory leak.
- After writing function check it for all scenarios.
- Try to avoid use of extra pointer variables.
- For each and every problem statement write separate main and separate function.
- All functions should return integer (Boolean). If function is successful then return 1 otherwise return 0.
- Follow prototype of function which is provided after problem statement.
- *For solving data structure problem first decide algorithm then draw memory representation diagram on paper and then start writing code.*

- *After writing every function check your function by calling such function which contains linked list traversals like Display() because this function detects unreferenced pointers and avoid segmentation fault.*

Demo Program :

```
///////////////////////////////////////////////////////////////////////////////////////////////////
/
//
// Write a program which insert new node at first position
//
///////////////////////////////////////////////////////////////////////////////////////////////////
/
// Algoritham
//
// START
//
// Accept element to be inserted
// Accept address of first node of linked list as head
//
// Allocate memory for new node as newnode
// Initialize the newly created node with input data
//
// if head is NULL
// head = newnode
//
// else
// store address of exiting node in next pointer of newnode
// newnode become head of linked list
//
// STOP


// Structure of linked list node
struct node
{
int data; // Data element
struct node*next; // Pointer to next node
};
```

```
///////////////////////////////////////////////////////////////////////////////////
/
//
// Function Name : InsertFirst
// Parameters :
// [IN/OUT] struct node **first.
// Holds head pointer of list.
//
// [IN]int no.
// Holds the entry which is to be added.
//
// Description : This function adds node to list.
//
// Returns : int
// If the function succeeds, the return value is 1.
// If the function fails, the return value is -1.
//
////////////////////////////////////////////////////////////////////////////////////

int
InsertFirst (
		struct node **first, // Address of first node
		int no // it is data element
		)
{
	struct node *newnode = NULL;

	//
	// Allocate new node.
	//
	newnode = (struct node *)malloc(sizeof(struct node));
	if (NULL == newnode) // Check whether memory is allocated or not
	{
		return -1;
	}

	//
	// Fill the node with data.
	//
	memset(newnode, 0, sizeof(struct node));
	newnode->next = NULL;
	newnode->data = no;

	// If linked list is empty
	if (NULL == *first)
	{
		*first = newnode;
	}
```

```
        // If linked list contains atleast one node
        else
        {
                newnode -> next = *first;
                *first = newnode;
        }
        return 1;
}

////////////////////////////////////////////////////////////////////////////////////////
/
//
// Function Name : DisplayList
// Parameters :
// [IN] struct node *first.
// Holds head pointer of list.
//
// Description : This function displays linked list.
//
// Returns : void
//
////////////////////////////////////////////////////////////////////////////////////////

void
DisplayList(
                struct node *first
                )
{

        // Loop used to traverse linked list

        while(first != NULL)
        {
                printf("| %d |->",first->data);
                first = first -> next;
        }
}

int main()
{
        // Pointer which hold address of linked list
        struct node *head = NULL;

        InsertFirst(&head,10);
        InsertFirst(&head,20);

        // Call this function which is used to display the contents
        DisplayList(head);

        return 0;
}
```

Write the following programs

Programs on singly linear linked list:

Linked list structure :

```
struct node
{
        int no;                     // Data element
        struct node *next,          // Address of next node
};
```

1. Write a program which add new node in singly linear linked list at first position.

```
        Function Prototype : int InsertFirst(
                                                struct node **Head,
                                                int no
                                          );
        Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
        Input data element : 21
        Output linked list : |21|->|10|->|20|->|30|->|40|->|50|->|60|->|70|
```

2. Write a program which add new node in singly linear linked list at last position.

```
        Function Prototype : int InsertLast(
                                                struct node **Head,
                                                int no
                                          );
        Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
        Input data element : 21
        Output linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|->|21|
```

3. Write a program which add new node in singly linear linked list at specified position.

```
Function Prototype : int InsertAtPosition(
                                                struct node **Head,
                                                int no,
                                                int pos
                                                );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Input data element : 21 Input position : 4
Output linked list : |10|->|20|->|30|->|21|->|40|->|50|->|60|->|70|
```

4. Write a program which remove first node from singly linear linked list.

```
Function Prototype : int DeleteFirst( struct node **Head );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Output linked list : |20|->|30|->|40|->|50|->|60|->|70|
```

5. Write a program which remove last node from singly linear linked list.

```
Function Prototype : int DeleteLast( struct node **Head );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Output linked list : |10|->|20|->|30|->|40|->|50|->|60|
```

6. Write a program which remove node from singly linear linked list which is at specified position.

```
Function Prototype : int DeleteAtPosition(
                                              struct node **Head,
                                              int pos
                                         );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Input position : 4
Output linked list : |10|->|20|->|30|->|50|->|60|->|70|
```

7. Write a program which displays contents of singly linear linked list.

```
Function Prototype : int Display( struct node *Head );
```

8. Write a program which search first occurrence of particular element from singly linear linked list. Function should return position at which element is found.

```
Function Prototype : int SearchFirstOcc(
                                              struct node *Head
                                              int no
                                         );
Input linked list : |10|->|20|->|30|->|40|->|50|->|30|->|70|
Input element : 30 Output : 3
```

9. Write a program which search last occurrence of particular element from singly linear linked list. Function should return position at which element is found.

```
Function Prototype : int SearchLastOcc(
                                              struct node *Head
                                              int no
                                         );
Input linked list : |10|->|20|->|30|->|40|->|50|->|30|->|70|
Input element : 30 Output : 6
```

10.Write a program which search second last occurrence of particular element from singly linear linked list. Function should return position at which element is found.

```
Function Prototype : int SearchSecLastOcc(
                                    struct node *Head
                                    int no
                                    );
Input linked list : |10|->|20|->|30|->|40|->|30|->|30|->|70|
Input element : 30 Output : 5
```

11.Write a program which searches all occurrence of particular element from singly linear linked list. Function should return number of occurance of that element.

```
Function Prototype : int SearchAll(
                               struct node *Head
                               int no
                               );
Input linked list : |10|->|20|->|30|->|40|->|30|->|30|->|70|
Input element : 30 Output : 3
```

12.Write a program which accept two singly linear linked list from user and concat source linked list after destination linked list.

```
Function Prototype : int ConcatList(
                               struct node **Src
                               struct node **Dest
                          );
Input source linked list : |30|->|30|->|70|
Input destination linked list : |10|->|20|->|30|->|40|
Output destination linked list : |10|->|20|->|30|->|40|->|30|->|30|-
>|70|
```

13.Write a program which accept two singly linear linked list from user and concat first N elements of source linked list after destination linked list.

```
Function Prototype : int ConcatFirstN(
                               struct node **Src
                               struct node **Dest
                               int no
                               );
Input source linked list : |30|->|30|->|70|
Input destination linked list : |10|->|20|->|30|->|40|
Input number of elements : 2
Output destination linked list : |10|->|20|->|30|->|40|->|30|->|30|
```

14.Write a program which accepts two singly linear linked list from user and concat last N elements of source linked list after destination linked list.

Function Prototype : int ConcatLastN(struct node **Src
struct node **Dest
int no);
Input source linked list : |30|->|30|->|70|

15.Write a program which accepts two singly linear linked list from user and also accept range and concat elements of source singly linear linked list from that range after singly linear destination linked list.

```
Function Prototype : int ConcatListRange(
                                         struct node **Src
                                         struct node **Dest
                                         int start
                                         int end
                                         );
Input source linked list : |30|->|30|->|70|->|80|->|90|->|100|
Input destination linked list : |30|->|40|
Input starting range : 2 Input ending range : 5
Output destination linked list : |30|->|40|->|30|->|70|->|80|->|90|
```

16.Write a program which copies contents of source singly linear linked list to singly linear destination linked list.

```
Function Prototype : int LLCopy(
                                    struct node **Src
                                    struct node **Dest
                                );
Input source linked list : |30|->|30|->|70|->|80|->|90|->|100|
Input destination linked list : Empty (NULL)
Output destination linked list : |30|->|30|->|70|->|80|->|90|->|100|
```

17.Write a program which copies first N contents of singly linear source linked list to destination singly linear linked list.

```
Function Prototype : int LLNCopy(
                                    struct node **Src
                                    struct node **Dest
                                    int no
                                    );
Input source linked list : |30|->|30|->|70|->|80|->|90|->|100|
Input destination linked list : Empty (NULL) Input no : 4
Output destination linked list : |30|->|30|->|70|->|80|
```

18.Write a program which copies last N contents of source singly linear linked list to destination singly linear linked list.

```
Function Prototype : int LLINCopy(
                                    struct node **Src
                                    struct node **Dest
                                    int no
                                    );
```

Input source linked list : |30|->|30|->|70|->|80|->|90|->|100|
Input destination linked list : Empty (NULL) Input no : 4

Output destination linked list : |70|->|80|->|90|->|100|

19.Write a program which copies contents of source singly linear linked list to destination singly linear linked list which lies between the particular range which is accepted from user.

```
Function Prototype : int LLCopyRange(
                                        struct node **Src
                                        struct node **Dest
                                        int start
                                        int end
                                  );
```

Input source linked list : |30|->|30|->|70|->|80|->|90|->|100|
Input destination linked list : Empty (NULL)
Input starting range : 2 Input starting range : 5
Output destination linked list : |30|->|70|->|80|->|90|

20.Write a program which copies alternate contents of source singly linear linked list to destination singly linear linked list.

```
Function Prototype : int LLCopyAlt(
                                  struct node **Src
                                  struct node **Dest
                            );
```

Input source linked list : |30|->|30|->|70|->|80|->|90|->|100|->|110|
Input destination linked list : Empty (NULL)
Output destination linked list : |30|->|70|->|90|->|110|

21.Write a program which copies contents of source singly linear linked list whose addition of digits is even number to destination singly linear linked list.

```
      Function Prototype : int LLCopyEx(
                                        struct node **Src
                                        struct node **Dest
                                        );
      Input source linked list : |30|->|33|->|73|->|80|->|90|->|100|->|110|
      Input destination linked list : Empty (NULL)
      Output destination linked list : |33|->|77|->|110|
```

22.Write a program which copies contents of source singly linear linked list whose addition of digits is prime number to destination singly linear linked list.

```
      Function Prototype : int LLCopyEx(
                                        struct node **Src
                                        struct node **Dest
                                  );
      Input source linked list : |30|->|32|->|73|->|80|->|70|->|110|->|112|
      Input destination linked list : Empty (NULL)
      Output destination linked list : |30|->|30|->|70|
```

23.Write a program which accept source singly linear linked list and destination singly linear linked list and check whether source list is sub list of destination list. Function returns first position at which sub list found.

```
Function Prototype : int SubList(
                                        struct node **Src
                                        struct node **Dest
                                        );
Input source linked list : |73|->|80|->|70|
Input dest list:|10|->|73|->|80|->|17|->|22|->|73|->|80|->|70|-|21|
Output : First Sub list found at position 6
```

24.Write a program which accept source singly linear linked list and destination singly linear linked list and check whether source list is sub list of destination list. Function returns last position at which sub list found.

```
Function Prototype : int SubList(
                                        struct node **Src
                                        struct node **Dest
                                        );
Input source linked list : |73|->|80|->|70|
Input dest list:|10|->|73|->|80|->|70|->|22|->|73|->|80|->|70|-|21|
Output : Last Sub list found at position 6
```

25.Write a program which accept source singly linear linked list from user and copy the contents into destination singly linear linked in ascending order.

```
Function Prototype : int CopyAsc(
                                        struct node **Src
                                        struct node **Dest
                                  );
Input source linked list : |110|->|73|->|10|->|80|->|70|->|12|
Input destination linked list: Empty (NULL)
Output destination linked list: |10|->|12|->|70|->|73|->|80|->|110|
```

26.Write a program which accept source singly linear linked list from user and copy the contents into destination singly linear linked in descending order.

```
Function Prototype : int CopyDsc(
                                        struct node **Src
                                        struct node **Dest
                                  );
Input source linked list : |110|->|73|->|10|->|80|->|70|->|12|
Input destination linked list: Empty (NULL)
Output destination linked list: |110|->|80|->|73|->|70|->|12|->|10|
```

27.Write a program which accept source singly linear linked list from user and reverse that linear linked.

Function Prototype : int ListReverse(struct node **Src);
Input source linked list : |110|->|73|->|10|->|80|->|70|->|12|
Output source linked list: |12|->|70|->|80|->|10|->|73|->|110|

28.Write a program which accept singly linear linked list from user and check whether list contains loop or not.

Function Prototype : int LoopChk(struct node *Src);

29.Write a program which accept two singly linear linked list and then merge this two list in such a way that resultant may contain elements alternatively from one list and other list.

Input First linked list : |10|->|20|->|30|->|40|
Input second linked list: |50|->|60|->|70|->|80|
Output linked list: |10|->|50|->|20|->|60|->|30|->|70|->|40|->|80|

30.Write a program which accept singly linear linked list and swap consecutive elements from that linked list.

Function Prototype : int Swap(struct node **Src);
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|->|80|->|90|
Output linked list :|20|->|10|->|40|->|30|->|60|->|50|->|80|->|70|->|90|

31.Write a program which accept source and destination linked list from user and we have to insert source linked list in destination linked list at given position.

Function Prototype : int InsertAt(struct node **Src
struct node **dest
int pos);
Input src linked list : |10|->|20|->|30|->|40|
Input dest linked list: |50|->|60|->|70|->|80|
Input position : 3
Output dest linked list: |50|->|60|->|10|->|20|->|30|->|40|->|70|->|80|

32.Write a program which accept singly linear linked list from user and reverse the linked list from particular range.

Function Prototype : int ListRevRange(
struct node **Src
int start,
int end
);

```
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Input starting position : 3 Input starting position : 6
Input linked list : |10|->|20|->|60|->|50|->|40|->|30|->|70|
```

33.Write a program which accepts two singly linear linked list from user and check whether they are intersecting or not.

```
Function Prototype : int ListIntersect(
                                        struct node *First
                                        struct node *Second
                                        );
```

34.Write a program which accept singly linear linked list from and check whether linked list is pallindrom or not.

```
Function Prototype : int ListPallindrom( struct node *First );
Input linked list : |10|->|20|->|30|->|40|->|30|->|20|->|10|
Output : The given linked list is pallindrom.
```

35.Write a program which accept singly linear linked list from user and sort that linked list in ascending order.

```
Function Prototype : int ListSortAsc( struct node **First );
Input linked list : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Output linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
```

36.Write a program which accept singly linear linked list from user and sort that linked list in descending order.

```
Function Prototype : int ListSortDec( struct node **First );
Input linked list : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Output linked list : |70|->|60|->|50|->|40|->|30|->|20|->|10|
```

37.Write a program which accepts two singly linear linked list from user and compare both linked list.

```
Function Prototype : int ListCmp(
                                        struct node *First
                                        struct node *Second
                                 );
Input linked list 1 : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Input linked list 2 : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Output linked list : Both the linked list are equal.
```

38.Write a program which accepts two singly linear linked list from user and compare both linked list till first N nodes.

```
Function Prototype : int ListNCmp(
                                        struct node *First
                                        struct node *Second
                                        int N);
```

Input linked list 1 : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Input linked list 2 : |60|->|70|->|50|->|40|->|80|->|30|->|70|
Input N : 4
Output linked list : Both the linked list are equal till node 4.

39.Write a program which accepts two singly linear linked list from user and compare both linked list but only last N nodes.

```
Function Prototype : int ListICmp(
                                        struct node *First
                                        struct node *Second
                                        int N
                                        );
```

Input linked list 1 : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Input linked list 1 : |40|->|80|->|70|->|60|->|30|->|10|->|20|
Input N : 3
Output linked list : Both the linked list are equal.

40.Write a program which accepts singly linear linked list from user and print middle node of list.

Function Prototype : int ListMid(struct node *First);
Input linked list : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Output : 40

Programs on singly Circular linked list:

Linked list structure :

```
struct node
{
        int no;                     // Data element
        struct node *next,          // Address of next node
};
```

Difference between linear and circular linked list is that last node of circular linked list holds address of first node.
Functions of linear and circular linked list is almost similar logically but there are some changes if

- Function requires list traversal because terminating conditions of loop is different.

- If function is going to add or delete the node at first or last position in circular linked list then there is some extra manipulation required otherwise functions of linear linked list are similar.

41.Write a program which add new node in singly circular linked list at first position.

```
Function Prototype : int InsertFirst(
                                    struct node **Head,
                                    struct node **Tail,
                                    int no );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Input data element : 21
Output linked list : |21|->|10|->|20|->|30|->|40|->|50|->|60|->|70|
```

42.Write a program which add new node in singly circular linked list at last position.

```
Function Prototype : int InsertLast(
                                    struct node **Head,
                                    struct node **Tail,
                                    int no
                                    );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Input data element : 21
Output linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|->|21|
```

43.Write a program which add new node in singly circular linked list at specified position.

```
Function Prototype : int InsertAtPosition(
                                        struct node **Head,
                                        struct node **Tail,
                                        int no,
                                        int pos );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Input data element : 21 Input position : 4
Output linked list : |10|->|20|->|30|->|21|->|40|->|50|->|60|->|70|
```

44.Write a program which remove first node from singly circular linked list.

```
Function Prototype : int DeleteFirst(
                                    struct node **Head,
                                    struct node **Tail
                                    );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Output linked list : |20|->|30|->|40|->|50|->|60|->|70|
```

45.Write a program which remove last node from singly circular linked list.

```
Function Prototype : int DeleteLast(
                                    struct node **Head
                                    struct node **Tail
                                    );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Output linked list : |10|->|20|->|30|->|40|->|50|->|60|
```

46.Write a program which remove node from singly circular linked list which is at specified position.

```
Function Prototype : int DeleteAtPosition(
                                        struct node **Head,
                                        struct node **Tail,
                                        int pos
                                        );
Input linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
Input position : 4
Output linked list : |10|->|20|->|30|->|50|->|60|->|70|
```

47.Write a program which displays contents of singly circular linked list.

```
Function Prototype : void Display( struct node *Head
struct node *Tail);
```

48.Write a program which search first occurrence of particular element from singly linear linked list. Function should return position at which element is found.

```
Function Prototype : int SearchFirstOcc(
                                        struct node *Head,
                                        struct node *Tail,
                                        int no );
Input linked list : |10|->|20|->|30|->|40|->|50|->|30|->|70|
Input element : 30 Output : 3
```

49.Write a program which search last occurrence of particular element from singly circular linked list. Function should return position at which element is found.

```
Function Prototype : int SearchLastOcc(
                                        struct node *Head,
                                        struct node *Tail,
                                        int no );
Input linked list : |10|->|20|->|30|->|40|->|50|->|30|->|70|
Input element : 30 Output : 6
```

50.Write a program which search second last occurrence of particular element from singly circular linked list. Function should return position at which element is found.

```
Function Prototype : int SearchSecLastOcc(
                                        struct node *Head,
                                        struct node *Tail,
                                        int no );
Input linked list : |10|->|20|->|30|->|40|->|30|->|30|->|70|
Input element : 30 Output : 5
```

51.Write a program which searches all occurrence of particular element from singly circular linked list. Function should return number of occurrence of that element.

Function Prototype : int SearchAll(
struct node *Head,
struct node *Tail,
int no);
Input linked list : |10|->|20|->|30|->|40|->|30|->|30|->|70|
Input element : 30 Output : 3

52.Write a program which accept two singly circular linked list from user and concat source linked list after destination linked list.

Function Prototype : int ConcatList(
struct node **SrcHead
struct node **SrcTail,
struct node **DestHead,
struct node **DestTail);
Input source linked list : |30|->|30|->|70|
Input destination linked list : |10|->|20|->|30|->|40|
Output destination linked list : |10|->|20|->|30|->|40|->|30|->|30|->|70|

53.Write a program which accept two singly circular linked list from user and concat first N elements of source linked list after destination linked list.

Function Prototype : int ConcatFirstN(
struct node **SrcHead
struct node **SrcTail,
struct node **DestHead,
struct node **DestTail,
int no);
Input source linked list : |30|->|30|->|70|
Input destination linked list : |10|->|20|->|30|->|40|
Input number of elements : 2
Output destination linked list : |10|->|20|->|30|->|40|->|30|->|30|

54.Write a program which accept singly circular linked list from and check whether linked list is pallindrom or not.

Function Prototype : int ListPallindrom(
struct node *First,
struct node *Tail);
Input linked list : |10|->|20|->|30|->|40|->|30|->|20|->|10|
Output : The given linked list is pallindrom.

55.Write a program which accept singly linear linked list from user and sort that linked list in ascending order.

```
Function Prototype : int ListSortAsc(
                                        struct node **Head,
                                        struct node **Tail);
Input linked list : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Output linked list : |10|->|20|->|30|->|40|->|50|->|60|->|70|
```

56.Write a program which accept singly linear linked list from user and sort that linked list in descending order.

```
Function Prototype : int ListSortDec(
                                        struct node **Head,
                                        struct node **Tail);
Input linked list : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Output linked list : |70|->|60|->|50|->|40|->|30|->|20|->|10|
```

57.Write a program which accepts two singly linear linked list from user and compare both linked list.

```
Function Prototype : int ListCmp(
                                        struct node *SrcHead,
                                        struct node *SrcTail,
                                        struct node *DestHead,
                                        struct node *DestTail);
Input linked list 1 : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Input linked list 2 : |60|->|70|->|50|->|40|->|30|->|10|->|20|

Output : Both lists are equal.
```

58.Write a program which accepts two singly linear linked list from user and compare both linked list till first N nodes.

```
Function Prototype : int ListNCmp(
                                        struct node *SrcHead
                                        struct node *SrcTail,
                                        struct node *DestHead,
                                        struct node *DestTail
                                        int N);
Input linked list 1 : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Input linked list 2 : |60|->|70|->|50|->|40|->|80|->|30|->|70|
Input N : 4
Output linked list : Both the linked list are equal till node 4.
```

59.Write a program which accepts two singly linear linked list from user and compare both linked list but only last N nodes.

```
Function Prototype : int ListICmp(
                                        struct node *SrcHead
                                        struct node *SrcTail,
                                        struct node *DestHead,
                                        struct node *DestTail
                                        int N);
```

Input linked list 1 : |60|->|70|->|50|->|40|->|30|->|10|->|20|
Input linked list 1 : |40|->|80|->|70|->|60|->|30|->|10|->|20|
Input N : 3
Output linked list : Both the linked list are equal.

Programs on Doubly Linear linked list:
Linked list structure :

```
struct node
{
        int no;                 // Data element
        struct node *next,      // Address of next node
        struct node *prev       // Address of next node
};
```

In doubly linked list we have to maintain address of previous node in the linked list.

60.Write a program which add new node in doubly linear linked list at first position.

```
Function Prototype : int InsertFirst(
struct node **Head,
int no );
```

Input linked list : |20|<=>|30|<=>|40|<=>|50|<=>|60|
Input data element : 21
Output linked list : |21|<=>|20|<=>|30|<=>|40|<=>|50|<=>|60|

61.Write a program which add new node in doubly linear linked list at last position.

```
Function Prototype : int InsertLast(
                                    struct node **Head,
                                    int no );
```

Input linked list : |20|<=>|30|<=>|40|<=>|50|<=>|60|<=>|70|
Input data element : 21
Output linked list
|20|<=>|30|<=>|40|<=>|50|<=>|60|<=>|70|<=>|21|

62.Write a program which add new node in doubly linear linked list at specified position.

```
Function Prototype : int InsertAtPosition(
                                          struct node **Head,
                                          int no,
                                          int pos );
```

Input linked list : |20|<=>|30|<=>|40|<=>|50|<=>|60|
Input data element : 21 Input position : 3
Output linked list :|20|<=>|30|<=>|21|<=>|40|<=>|50|<=>|60|

63.Write a program which remove first node from doubly linear linked list.

Function Prototype : int DeleteFirst(struct node **Head);
Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|
Output linked list : |20|<=>|30|<=>|21|<=>|40|<=>|50|

64.Write a program which remove last node from doubly linear linked list.

Function Prototype : int DeleteLast(struct node **Head);
Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|
Output linked list : |10|<=>|20|<=>|30|<=>|40|

65.Write a program which remove node from doubly linear linked list which is at specified position.

```
Function Prototype : int DeleteAtPosition(
                                         struct node **Head,
                                         int pos );
```

Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|
Input position : 4
Output linked list : |10|<=>|20|<=>|30|<=>|50|

66.Write a program which displays contents of doubly linear linked list.

Function Prototype : int Display(struct node *Head);

67.Write a program which search first occurrence of particular element from doubly linear linked list. Function should return position at which element is found.

```
Function Prototype : int SearchFirstOcc(
                                         struct node *Head
                                         int no );
```

Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|
Input element : 30 Output : 3

68.Write a program which search last occurrence of particular element from doubly linear linked list. Function should return position at which element is found.

```
Function Prototype : int SearchLastOcc(
                                         struct node *Head
                                         int no );
```

Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|<=>|30|
Input element : 30 Output : 6

69.Write a program which search second last occurrence of particular element from doubly linear linked list. Function should return position at which element is found.

```
        Function Prototype : int SearchSecLastOcc(
                                                  struct node *Head
                                                  int no );
        Input linked list : |20|<=>|30|<=>|40|<=>|30|<=>|30|<=>|70|
        Input element : 30 Output : 5
```

70.Write a program which searches all occurrence of particular element from doubly linear linked list. Function should return number of occurrence of that element.

```
        Function Prototype : int SearchAll(
                                           struct node *Head
                                           int no );
        Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|30|<=>|30|
        Input element : 30 Output : 3
```

71.Write a program which accept two doubly linear linked list from user and concat source linked list after destination linked list.

```
        Function Prototype : int ConcatList(
                                           struct node **Src
                                           struct node **Dest );
        Input source linked list : |30|<=>|70|
        Input destination linked list : |10|<=>|20|<=>|30|
        Output : |30|<=>|70|<=>|10|<=>|20|<=>|30|
```

72.Write a program which accept two doubly linear linked list from user and concat first N elements of source linked list after destination linked list.

```
        Function Prototype : int ConcatFirstN( struct node **Src
                                           struct node **Dest
                                           int no );
        Input source linked list : |30|<=>|30|<=>|70|
        Input destination linked list : |10|<=>|20|<=>|30|<=>|40|
        Input number of elements : 2
        Output destination list : |10|<=>|20|<=>|30|<=>|40|<=>|30|<=>|30|
```

73.Write a program which accepts two doubly linear linked list from user and concat last N elements of source linked list after destination linked list.

```
        Function Prototype : int ConcatLastN( struct node **Src
                                           struct node **Dest
                                           int no );
        Input source linked list : |30|<=>|30|<=>|70|
        Input destination linked list : |20|<=>|30|<=>|40|
        Input number of elements : 2
        Output destination linked list : |20|<=>|30|<=>|40|<=>|30|<=>|70|
```

74.Write a program which accepts two doubly linear linked list from user and also accept range and concat elements of source singly linear linked list from that range after doubly linear destination linked list.

```
Function Prototype : int ConcatListRange(     struct node **Src
                                              struct node **Dest,
                                              int start ,
                                              int end);
Input source linked list :
|30|<=>|30|<=>|70|<=>|80|<=>|90|<=>|100|
Input destination linked list : |30|<=>|40|
Input starting range : 2 Input ending range : 5
Output destination list:|30|<=>|40|<=>|30|<=>|70|<=>|80|<=>|90|
```

75.Write a program which copies contents of source doubly linear linked list to doubly linear destination linked list.

```
Function Prototype : int LLCopy( struct node **Src
                                 struct node **Dest);
```

76.Write a program which copies first N contents of doubly linear source linked list to destination doubly linear linked list.

```
Function Prototype : int LLNCopy(      struct node **Src
                                       struct node **Dest
                                       int no);
Input source linked list :
|30|<=>|30|<=>|70|<=>|80|<=>|90|<=>|100|
Input destination linked list : Empty (NULL) Input no : 4
Output destination linked list : |30|<=>|30|<=>|70|<=>|80|
```

77.Write a program which copies last N contents of source doubly linear linked list to destination doubly linear linked list.

```
Function Prototype : int LLINCopy(     struct node **Src
                                       struct node **Dest
                                       int no);
Input source linked list :
|30|<=>|30|<=>|70|<=>|80|<=>|90|<=>|100|
Input destination linked list : Empty (NULL) Input no : 4
Output destination linked list : |70|<=>|80|<=>|90|<=>|100|
```

78.Write a program which copies contents of source doubly linear linked list to destination doubly linear linked list which lies between the particular range which is accepted from user.

```
Function Prototype : int LLCopyRange(struct node **Src
                                     struct node **Dest
                                     int start
                                     int end);
Input source linked list :
|30|<=>|30|<=>|70|<=>|80|<=>|90|<=>|100|
Input destination linked list : Empty (NULL)
Input starting range : 2 Input starting range : 5
```

Output destination linked list : |30|<=>|70|<=>|80|<=>|90|

79.Write a program which copies alternate contents of source doubly linear linked list to destination doubly linear linked list.

```
Function Prototype : int LLCopyAlt(struct node **Src
                                   struct node **Dest);
Input src
list:|30|<=>|30|<=>|70|<=>|80|<=>|90|<=>|100|<=>|110|
Input destination linked list : Empty (NULL)
Output destination linked list : |30|<=>|70|<=>|90|<=>|110|
```

80.Write a program which copies contents of source doubly linear linked list whose addition of digits is even number to destination doubly linear linked list.

```
Function Prototype : int LLCopyEx(      struct node **Src
                                        struct node **Dest);
Input src list :
|30|<=>|33|<=>|73|<=>|80|<=>|90|<=>|100|<=>|110|
Input destination linked list : Empty (NULL)
Output destination linked list : |33|<=>|77|<=>|110|
```

81.Write a program which copies contents of source doubly linear linked list whose addition of digits is prime number to destination doubly linear linked list.

```
Function Prototype : int LLCopyEx(struct node **Src
                                        struct node **Dest);
Input src
list:|30|<=>|32|<=>|73|<=>|80|<=>|70|<=>|110|<=>|112|
Input destination linked list : Empty (NULL)
Output destination linked list : |30|<=>|30|<=>|70|
```

82.Write a program which accept source doubly linear linked list and destination doubly linear linked list and check whether source list is sub list of destination list. Function returns first position at which sub list found.

```
Function Prototype : int SubList(struct node **Src
                                 struct node **Dest);
Input source linked list : |73|<=>|80|<=>|70|
Input dest list: |80|<=>|17|<=>|22|<=>|73|<=>|80|<=>|70|
Output : First Sub list found at position 4
```

83.Write a program which accept source doubly linear linked list and destination doubly linear linked list and check whether source list is sub list of destination list. Function returns last position at which sub list found.

```
Function Prototype : int SubList(struct node **Src
                                 struct node **Dest);
Input source linked list : |73|<=>|80|<=>|70|
Input dest list:|73|<=>|80|<=>|70|<=>|22|<=>|73|<=>|80|<=>|70|
Output : Last Sub list found at position 5
```

84.Write a program which accept source doubly linear linked list from user and copy the contents into destination doubly linear linked in ascending order.

```
        Function Prototype : int CopyAsc(struct node **Src
                                         struct node **Dest);
        Input source linked list : |73|<=>|10|<=>|80|<=>|70|<=>|12|
        Input destination linked list: Empty (NULL)
        Output destination linked list: |10|<=>|12|<=>|70|<=>|73|<=>|80|
```

85.Write a program which accept source doubly linear linked list from user and copy the contents into destination doubly linear linked in descending order.

```
        Function Prototype : int CopyDsc(struct node **Src
                                         struct node **Dest);
        Input source linked list : |73|<=>|10|<=>|80|<=>|70|<=>|12|
        Input destination linked list: Empty (NULL)
        Output destination linked list: |80|<=>|73|<=>|70|<=>|12|<=>|10|
```

Programs on doubly Circular linked list:

Linked list structure :

```
struct node
{
        int no;                     // Data element
        struct node *next,          // Address of next node
        struct node *prev           // Address of next node
};
```

In doubly linked list next pointer of last node holds address of first node and previous pointer of first node holds address of last node.

86.Write a program which add new node in singly circular linked list at first position.

```
        Function Prototype : int InsertFirst(
                                                struct node **Head,
                                                struct node **Tail,
                                                int no );
        Input linked list : |30|<=>|40|<=>|50|<=>|60|<=>|70|
        Input data element : 21
        Output linked list : |21|<=>|30|<=>|40|<=>|50|<=>|60|<=>|70|
```

87.Write a program which add new node in singly circular linked list at last position.

```
        Function Prototype : int InsertLast(
                                                struct node **Head,
                                                struct node **Tail,
                                                int no );
        Input linked list : |30|<=>|40|<=>|50|<=>|60|<=>|70|
```

Input data element : 21
Output linked list : |30|<=>|40|<=>|50|<=>|60|<=>|70|<=>|21|

88.Write a program which add new node in singly circular linked list at specified position.

```
Function Prototype : int InsertAtPosition(
                                    struct node **Head,
                                    struct node **Tail,
                                    int no,
                                    int pos );
Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|<=>|60|
Input data element : 21 Input position : 4
Output linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|<=>|60|
```

89.Write a program which remove first node from singly circular linked list.

```
Function Prototype : int DeleteFirst( struct node **Head,
                              struct node **Tail);
Input linked list : |40|<=>|50|<=>|60|<=>|70|
Output linked list : |50|<=>|60|<=>|70|
```

90.Write a program which remove last node from singly circular linked list.

```
Function Prototype : int DeleteLast( struct node **Head
                              struct node **Tail);
Input linked list : |10|<=>|20|<=>|30|<=>|40|
Output linked list : |10|<=>|20|<=>|30|
```

91.Write a program which remove node from singly circular linked list which is at specified position.

```
Function Prototype : int DeleteAtPosition(
                                    struct node **Head,
                                    struct node **Tail,
                                    int pos );
Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|
```

Input position : 4
Output linked list : |10|<=>|20|<=>|30|<=>|50|

92.Write a program which displays contents of singly circular linked list.

```
Function Prototype : void Display( struct node *Head
                                   struct node *Tail);
```

93.Write a program which search first occurrence of particular element from singly linear linked list. Function should return position at which element is found.

```
Function Prototype : int SearchFirstOcc(    struct node *Head,
                                            struct node *Tail,
                                            int no );
Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|<=>|30|
Input element : 30 Output : 3
```

94.Write a program which search last occurrence of particular element from singly circular linked list. Function should return position at which element is found.

```
Function Prototype : int SearchLastOcc( struct node *Head,
                                        struct node *Tail,
                                        int no );
Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|50|<=>|30|
Input element : 30 Output : 6
```

95.Write a program which search second last occurrence of particular element from doubly circular linked list. Function should return position at which element is found.

```
Function Prototype : int SearchSecLastOcc(  struct node *Head,
                                            struct node *Tail,
                                            int no );
Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|30|<=>|30|
Input element : 30 Output : 5
```

96.Write a program which searches all occurrence of particular element from doubly circular linked list. Function should return number of occurance of that element.

```
Function Prototype : int SearchAll(     struct node *Head,
                                        struct node *Tail,
                                        int no );
Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|30|<=>|30|
Input element : 30 Output : 3
```

97.Write a program which accept two doubly circular linked list from user and concat source linked list after destination linked list.

```
Function Prototype : int ConcatList(     struct node **SrcHead
                                         struct node **SrcTail,
                                         struct node **DestHead,
                                         struct node **DestTail );
```

Input source linked list : |30|<=>|30|<=>|70|
Input destination linked list : |10|<=>|20|
Output destination linked list : |10|<=>|20|<=>|30|<=>|30|<=>|70|

98.Write a program which accept two doubly circular linked list from user and concat first N elements of source linked list after destination linked list.

```
Function Prototype : int ConcatFirstN( struct node **SrcHead
                                         struct node **SrcTail,
                                         struct node **DestHead,
                                         struct node **DestTail,
                                         int no );
```

Input source linked list : |30|<=>|30|<=>|70|
Input destination linked list : |30|<=>|40|
Input number of elements : 2
Output destination linked list : |30|<=>|40|<=>|30|<=>|30|

99.Write a program which accept doubly circular linked list from and check whether linked list is pallindrom or not.

```
Function Prototype : int ListPallindrom( struct node *First,
                                         struct node *Tail);
```

Input linked list : |10|<=>|20|<=>|30|<=>|40|<=>|30|<=>|20|
Output : The given linked list is pallindrom.

100. Write a program which accept doubly linear linked list from user and sort that linked list in ascending order.

```
Function Prototype : int ListSortAsc(    struct node **Head,
                                         struct node **Tail);
```

Input linked list : |60|<=>|70|<=>|50|<=>|40|<=>|30|<=>|10|
Output linked list : |10|<=>|30|<=>|40|<=>|50|<=>|60|<=>|70|