Experiment No. 3

Aim : Implementation of ~~singly linked list / circular~~ singly linked list and various operations for real-world.

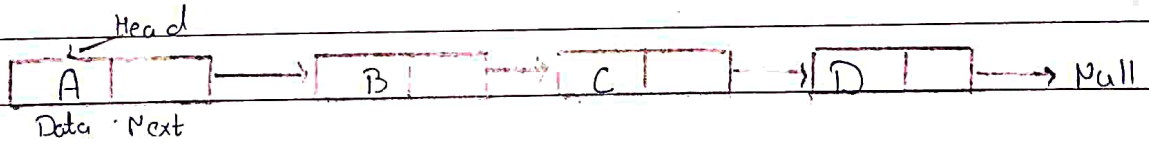
Objectives :

1) To use the basic principles of programming as applied to complex data structures.
2) To learn the principles of Linked List and its various operations.

Theory :

Introduction to linked list :

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers.



In simple words, a linked list consists of nodes where each node contains a data field & a reference to the next node in the list.

Singly linked List : It is the simplest type of linked list in which every node contains some data and a pointer to the next node of the same data type. The node contains a pointer to the next node means that the node stores the address of the next node in the sequence.

~~Introduction to circular~~

Insertion : The insertion into a singly linked list can be performed at different positions Based on the position of the new node being inserted, the insertion is categorized into the following categories :

1 Insertion at begineing - It involves inserting any element at the front of the list.

2. Insertion at the end of the list - It involves insertion at the last of the linked list. The new node can be inserted as the only node in the list or it can be inserted as the last node.

3. Insertion after specific node : It involves insertion after the specified node of the linked list. We need to skip the desired number of nodes in order to reach the node after which the new node will be inserted.

Deletion : The deletion of a node from a singly linked list can be perfored at different position. Based on the position of node being deleted, the operation is categorized as :

1 Deletion at beginning : It involves deletion of a ~~bad~~ node from the beginning of the list.

2 Deletion at end : It involves deleting the last node of the list

3 Deletion after specified node : It involes deleting the node after the specified node in the list.

Traversing : In traversing, we simply visit each node of the list at least onee in order to perform some specific operation on it.

Algorithm:

Insertion in the beginning

Step 1: IF PTR = NULL
write OVERFLOW
GO to step 7
[END of IF]

Step 2: SET, NEW_NODE = PTR

Step 3: SET PTR = PTR -> NEXT

Step 4: SET NEW_NODE -> DATA = VAL

Step 5: SET NEW_NODE -> NEXT = HEAD

Step 6: SET HEAD = NEW_NODE

Step 7: EXIT

Insertion at the End.

Step 1: IF PTR = NULL Write Overflow
GO to step 10
[END OF IF]

Step 2: SET NEW_NODE = PTR

Step 3: SET PTR = PTR -> NEXT

Step 4: SET NEW_NODE -> DATA = VAL

Step 5: SET NEW_NODE -> NEXT = NULL

Step 6: SET PTR = HEAD

Step 7: Repeat step 8 while PTR -> NEXT != NULL

Step 8: SET PTR = PTR -> NEXT
[END OF LOOP]

Step 9: SET PTR -> NEXT = NEW_NODE

Step 10: EXIT.

Insertion at specified node.

Step 1 : IF PTR = NULL
WRITE OVERFLOW
GO TO Step 12
END OF IF
Step 2 : SET NEW-NODE = PTR
Step 3 : NEW-NODE → DATA = VAL
Step 4 : SET TEMP = HEAD
Step 5 : SET I = 0
Step 6 : REPEAT STEP 5 and 6 until I
Step 7 : TEMP = TEMP → NEXT
Step 8 : IF TEMP = NULL
~~step~~ WRITE "DESIRED NODE NOT PRESENT"
GO TO STEP 12
END OF IF
END OF LOOP
Step 9 : PTR → NEXT = TEMP → NEXT
Step 10 : TEMP → NEXT = PTR
Step 11 : SET PTR = NEW-NODE
Step 12 : EXIT

Deletion at specified node

Step 1 : IF HEAD = NULL
WRITE UNDERFLOW
GOTO STEP 10
END OF IF
Step 2 : SET TEMP = HEAD

Step 3: SET I = 0

Step 4 : REPEAT STEP, 5 TO 8 UNTIL I

Step 5: TEMP 1 = TEMP

Step 6 : TEMP = TEMP -> NEXT

Step 7 : IF TEMP = NULL

WRITE " DESIRED NODE NOT PRESENT "

GO TO STEP 11

END OF IF

Step 8: I = I + 1

END OF LOOP

Step 9: TEMP 1 -> NEXT = TEMP -> NEXT

Step 10 : FREE TEMP

Step 11 : EXIT

Deletion at beginning:

Step 1: IF HEAD = NULL

Write UNDERFlow

GO to Step 5

[END OF IF]

Step 2: SET PTR = HEAD

Step 3: SET HEAD = HEAD -> NEXT

Step 4: FREE PTR

Step 5: EXIT

Deletion at END :

Step 1 : IF HEAD = NULL

Write UNDERFLOW

Go to step 8
[END OF IF]
Step 2: SET PTR = HEAD
Step 3: Repect steps 4 and 5 while PTR → NEXT !=
Step 4: SET PREPTR = PTR
Step 5: SET PTR = PTR → NEXT
[END OF LOOP]
Step 6: SET PREPTR → NEXT = NULL
Step 7: FREE PTR
Step 8: EXIT.

Examples:
1) List of users of a website that need to be emailed some notification.
2) List of objects in a 3D game that need to b rendered to the screen.

Conclusion: Thus, we have studied the concepts & implementation of singly linked list & its various operations. ~~Use~~ We learn use of ~~singl~~ Linked list in real-world applications.

Outcome: Apply the concepts of singly linked list for real-world applications.

PROGRAM:

```
/*****************************************************************************
  Menu-driven Program for implementation of Singly Linked list
*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>

// Defining Structure
typedef struct node
{
    int data;
    struct node *next;
} node;
node *createList();
node *Insert_beg(node *head, int x);
node *Insert_end(node *head, int x);
node *Insert_mid(node *head, int x);
node *Delete_beg(node *head);
node *Delete_end(node *head);
node *Delete_mid(node *head);
void PrintList(node *head);

// Main Function
void main()
{
    int choice, insert_option, delete_option, x;
    node *head = NULL;
    printf("Welcome to the implementation of the singly linked list..\n");
    do
    {
        printf("\nLinked list Operation available: \n");
        printf("1.Create a List \n");
        printf("2.Insert a node in list \n");
        printf("3.Delete a node from list \n");
        printf("4.Display existing list \n");
        printf("5.Exit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            head = createList();
            break;
        case 2:
            do
            {
                printf("Select a position where you to want to insert new nod
                printf(" 1. Beginning of the List \n");
                printf(" 2. At the end of the list \n");
                printf(" 3. Insert in between \n");
                printf(" 4. Exit the insert operation \n");
                printf("Enter your choice: ");
                scanf("%d", &insert_option);
                switch (insert_option)
                {
                case 1:
                    printf("Enter the data to be inserted: ");
                    scanf("%d", &x);
                    head = Insert_beg(head, x);
                    break;
                case 2:
                    printf("Enter the data to be inserted: ");
                    scanf("%d", &x);
                    head = Insert_end(head, x);
                    break;
                case 3:
                    printf("Enter the data to be inserted: ");
                    scanf("%d", &x);
                    head = Insert_mid(head, x);
                    break;
                case 4:
                    printf("Insert operation Exit");
                    break;
                default:
                    printf("Please enter a valid choice: 1, 2, 3, 4");
                }
            } while (insert_option != 4);
            printf("\n \n");
            break;
```

```
                case 3:
                    printf("Enter the data to be inserted: ");
                    scanf("%d", &x);
                    head = Insert_mid(head, x);
                    break;
                case 4:
                    printf("Insert operation Exit");
                    break;
                default:
                    printf("Please enter a valid choice: 1, 2, 3, 4");
                }
            } while (insert_option != 4);
            printf("\n \n");
            break;
        case 3:
            do
            {
                printf("Select a position from where you to want to delete the
                printf(" 1. Beginning of the List \n");
                printf(" 2. At the end of the list \n");
                printf(" 3. From specified node \n");
                printf(" 4. Exit the delete operation \n");
                printf("Enter your choice: ");
                scanf("%d", &delete_option);
                switch (delete_option)
                {
                case 1:
                    head = Delete_beg(head);
                    break;
                case 2:
                    head = Delete_end(head);
                    break;
                case 3:
                    head = Delete_mid(head);
                    break;
                case 4:
                    printf("Delete Operation Exit");
                    break;
                default:
                    printf("Please enter a valid choice: 1, 2, 3, 4");
                }
            } while (delete_option != 4);
            printf("\n \n");
            break;
        case 4:
            PrintList(head);
            break;
        case 5:
            printf("Exit: Program Finished !!");
            break;
        default:
            printf("Please enter a valid choice: 1, 2, 3, 4, 5");
        }
    } while (choice != 5);
}

// Function to create List
node *createList()
{
    node *head, *p;
    int i, n;
    head = NULL;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    printf("Enter the data: ");
    for (i = 0; i <= n - 1; i++)
    {
        if (head == NULL)
        {
            p = head = (node *)malloc(sizeof(node));
        }
        else
        {
            p->next = (node *)malloc(sizeof(node));
            p = p->next;
        }
        p->next = NULL;
        scanf("%d", &(p->data));
    }
    printf("\n \n");
    return (head);
}

// Function to insert element
node *Insert_beg(node *head, int x)
{
    node *p;
    p = (node *)malloc(sizeof(node));
    p->data = x;
    p->next = head;
    head = p;
    return (head);
}
node *Insert_end(node *head, int x)
{
    node *p, *q;
    p = (node *)malloc(sizeof(node));
    p->data = x;
    p->next = NULL;
    if (head == NULL)
        return (p);
    for (q = head; q->next != NULL; q = q->next)
        ;
    q->next = p;
    return (head);
}
```

```
node *Insert_mid(node *head, int x)
{
    node *p, *q;
    int y;
    p = (node *)malloc(sizeof(node));
    p->data = x;
    p->next = NULL;
    printf("After which element you want to insert the new element ?");
    scanf("%d", &y);
    for (q = head; q != NULL && q->data != y; q = q->next)
        ;
    if (q != NULL)
    {
        p->next = q->next;
        q->next = p;
    }
    else
        printf("ERROR !! Data Not Found");
    return (head);
}

// Function to delete element
node *Delete_beg(node *head)
{
    node *p, *q;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
    }
    p = head;
    head = head->next;
    free(p);
    return (head);
}
node *Delete_end(node *head)
{
    node *p, *q;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
    }
    p = head;
    if (head->next == NULL)
    {
        head = NULL;
        free(p);
        return (head);
    }
    for (q = head; q->next->next != NULL; q = q->next)
        p = q->next;
    q->next = NULL;
    free(p);
    return (head);
}
node *Delete_mid(node *head)
{
    node *p, *q;
    int x, i;
    if (head == NULL)
    {
        printf("Empty Linked List");
        return (head);
    }
    printf("Enter the data to be deleted: ");
    scanf("%d", &x);
    if (head->data == x)
    {
        p = head;
        head = head->next;
        free(p);
        return (head);
    }
    for (q = head; q->next->data != x && q->next != NULL; q = q->next)
        if (q->next == NULL)
        {
            printf("ERROR !! Data Not Found");
            return (head);
        }
    p = q->next;
    q->next = q->next->next;
    free(p);
    return (head);
}

// Function to print the existing list
void PrintList(node *head)
{
    node *p;
    printf("[ ");
    for (p = head; p != NULL; p = p->next)
    {
        printf("%d \t", p->data);
    }
    printf(" ]");
    printf("\n \n");
}
```

OUTPUT:

```
C:\TURBOC3\BIN>TC
Welcome to the implementation of the singly linked list..
Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 1
Enter the number of nodes: 3
Enter the data: 1
2
3


Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 4
[ 1     2       3       ]
Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 3
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. From specified node
 4. Exit the delete operation
Enter your choice: 1
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. From specified node
 4. Exit the delete operation
Enter your choice: 4
Delete Operation Exit

Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 4
[ 2     3       ]

Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 3
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. From specified node
 4. Exit the delete operation
Enter your choice: 2
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. From specified node
 4. Exit the delete operation
Enter your choice: 4
Delete Operation Exit

Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 4
[ 2     ]
```

```
Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 3
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. From specified node
 4. Exit the delete operation
Enter your choice: 3
Enter the data to be deleted: 2
Select a position from where you to want to delete the element
 1. Beginning of the List
 2. At the end of the list
 3. From specified node
 4. Exit the delete operation
Enter your choice: 4
Delete Operation Exit

Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 4
[  ]
Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 2
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 1
Enter the data to be inserted: 1
Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 4
[ 1      ]

Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 2
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 2
Enter the data to be inserted: 2
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 4
Insert operation Exit

Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 4
[ 1     2        ]
```

```
Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 2
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 3
Enter the data to be inserted: 3
After which element you want to insert the new element ? 1
Select a position where you to want to insert new node
 1. Beginning of the List
 2. At the end of the list
 3. Insert in between
 4. Exit the insert operation
Enter your choice: 4
Insert operation Exit

Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 4
[ 1     3       2       ]

Linked list Operation available:
1.Create a List
2.Insert a node in list
3.Delete a node from list
4.Display existing list
5.Exit
Enter your choice : 5_
```