**2.Linked List**

Problem Statement:

Write C Program to show the different operations on a singly linked list. Let the options 0-10 indicates the following

operations:

O: Exit

1: Insert a node in the beginning of the linked list

2. Insert a node at the end of the linked list

3. Insert a node at the kth position of the linked list

4. Search a node containing a given value

5. Print the entire linked list

6. Delete a node from the beginning of the linked list

7. Dele a node at the end of the linked list

8.Delete a node at the kth position of the linked list

9. Continue

The user should be able to give his/her option to perform different operations on a singly linked list using the above

menu as long as he/ she wants.

• Output example :

Type 1 to insert at first

Type 2 to insert at last

Type 3 to insert at any pos

Type 4 to search

Type 5 to print the list

Type 6 to delete the first element

Type 7 to delete at last position

Type 8 to delete at a given position

TYpe 0 to terminate

1

Enter value: 5

2

Enter value: 6

5

5 ->6 ->

3

Enter value & position: 7 1

5

5 ->7 ->6 ->

1

Enter value: 56

1

Enter value: 28

5

28 ->56 ->5 ->7 ->6 ->

4

Enter value: 7

Index at: 3

6

5

56 ->5 ->7 ->6 ->

7

5

56 ->5 ->7 ->

8

Enter position: 1

5

56 ->7 ->

0

Terminated

Proposed C Code:

/\* ------- main.c ------- \*/

#include <stdio.h>

#include <stdlib.h>

typedef struct N

{

    int data;

    struct N \*next;

} Node;

Node \*insertatfirst(Node \*head, int val)

{

    Node \*temp;

    temp = (Node \*)malloc(sizeof(Node));

    temp->data = val;

    if (head == NULL)

    {

        temp->next = NULL;

    }

    else

    {

        temp->next = head;

    }

    head = temp;

    return head;

}

Node \*insertatlast(Node \*head, int val)

{

    Node \*temp;

    temp = (Node \*)malloc(sizeof(Node));

    temp->data = val;

    temp->next = NULL;

    Node \*p = head;

    while (p->next != NULL)

    {

        p = p->next;

    }

    p->next = temp;

    return head;

}

Node \*insertatpos(Node \*head, int val, int pos)

{

    if (pos == 0)

    {

        head = insertatfirst(head, val);

        return head;

    }

    Node \*temp;

    temp = (Node \*)malloc(sizeof(Node));

    temp->data = val;

    temp->next = NULL;

    int i = pos - 1;

    Node \*p = head;

    while (i > 0)

    {

        p = p->next;

        i--;

    }

    Node \*new = p->next;

    p->next = temp;

    temp->next = new;

    return head;

}

int search(Node \*head, int val)

{

    Node \*temp = head;

    int i = 0;

    while (temp != NULL)

    {

        if (temp->data == val)

        {

            return i;

        }

        i++;

        temp = temp->next;

    }

    return -1;

}

Node \*deleteatfirst(Node \*head)

{

    head = head->next;

    return head;

}

Node \*deleteatlast(Node \*head)

{

    Node \*temp = head;

    while (temp->next->next != NULL)

    {

        temp = temp->next;

    }

    temp->next = NULL;

    return head;

}

Node \*deleteatpos(Node \*head, int pos)

{

    if (pos == 0)

    {

        head = deleteatfirst(head);

        return head;

    }

    Node \*temp = head;

    int i = pos - 1;

    while (i > 0)

    {

        temp = temp->next;

        i--;

    }

    temp->next = temp->next->next;

    return head;

}

void display(Node \*head)

{

    Node \*temp = head;

    while (temp != NULL)

    {

        printf("%d ->", temp->data);

        temp = temp->next;

    }

    printf("\n");

}

int main()

{

    Node \*head = NULL;

    printf("Type 1 to insert at first\nType 2 to insert at last \nType 3 to insert at any pos \nType 4 to search \nType 5 to print the list \nType 6 to delete the first element \nType 7 to delete at last position\nType 8 to delete at a given position\nTYpe 0 to terminate\n");

    int x, pos, val;

    do

    {

        scanf("%d", &x);

        switch (x)

        {

        case 0:

            break;

        case 1:

            printf("Enter value: ");

            scanf("%d", &val);

            head = insertatfirst(head, val);

            break;

        case 2:

            printf("Enter value: ");

            scanf("%d", &val);

            head = insertatlast(head, val);

            break;

        case 3:

            printf("Enter value & position: ");

            scanf("%d %d", &val, &pos);

            head = insertatpos(head, val, pos);

            break;

        case 4:

            printf("Enter value: ");

            scanf("%d", &val);

            printf("Index at: %d\n", search(head, val));

            break;

        case 5:

            display(head);

            break;

        case 6:

            head = deleteatfirst(head);

            break;

        case 7:

            head = deleteatlast(head);

            break;

        case 8:

            printf("Enter position: ");

            scanf("%d", &pos);

            head = deleteatpos(head, pos);

            break;

        }

    }while (x!=0);

    printf("Terminated");

    return 0;

}

/\* ---------------------- \*/

Conclusion:

The proposed algorithm has a runtime of O(n), where n is the length of the linked list.

Limitations and assumptions for this algorithm include:

1.Here insertion and deletion are take place at O(n) time in stead of constant time(O(1)).

2.Here we can move only in one direction, we cannot move in the previous direction.