

LAB 9: DLL

```
#include <stdio.h>

#include <stdlib.h>

struct node
{
    struct node *prev;
    int n;
    struct node *next;
}*h,*temp,*temp1,*temp2,*temp4;

void insert_beg();

void insert_atpos();
void display_beg();
void delete_atpos();

int count = 0;

int main()
{
    int ch;

    h = NULL;
    temp = temp1 = NULL;

    printf("\n 1 - Insert at Beginning");
```

```
printf("\n 2 - Delete At Particular Position");
printf("\n 3 - Display from Beginning");
printf("\n 4 - Exit");

while (1)
{
    printf("\n Enter choice :: ");
    scanf("%d", &ch);
    switch (ch)
    {
        case 1:
            insert_beg();
            break;
        case 2:
            delete_atpos();
            break;
        case 3:
            display_beg();
            break;
        case 4:
            exit(0);
        default:
            printf("\n Entered the Wrong choice from the menu!!!!!!!!!!!!!!");
    }
}
}
```

```

/* TO create an empty node */
void create()
{
    int data;

    temp =(struct node *)malloc(1*sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\n Enter value to node :: ");
    scanf("%d", &data);
    temp->n = data;
    count++;
}

```

```

/* TO insert at beginning */
void insert_beg()
{
    if (h == NULL)
    {
        create();
        h = temp;
        temp1 = h;
    }
    else
    {

```

```

        create();
        temp->next = h;
        h->prev = temp;
        h = temp;
    }
}

```

```

/* To delete an element */
void delete_atpos()
{
    int i = 1, pos;

    printf("\n Enter position to be deleted : ");
    scanf("%d", &pos);
    temp2 = h;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Error : Position out of range to delete!!!!!!");
        return;
    }
    if (h == NULL)
    {
        printf("\n Error : Empty list no elements to delete!!!!!!");
    }
}

```

```
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        if (i == 1)
        {
            if (temp2->next == NULL)
            {
                printf("Node deleted from list");
                free(temp2);
                temp2 = h = NULL;
                return;
            }
        }
        if (temp2->next == NULL)
        {
            temp2->prev->next = NULL;
            free(temp2);
            printf("Node deleted from list");
            return;
        }
    }
```

```

temp2->next->prev = temp2->prev;
if (i != 1)
    temp2->prev->next = temp2->next; /* Might not need this statement
if i == 1 check */
    if (i == 1)
        h = temp2->next;
        printf("\n Node deleted");
        free(temp2);
    }
    count--;
}

```

/* display from beginning */

void display_beg()

```

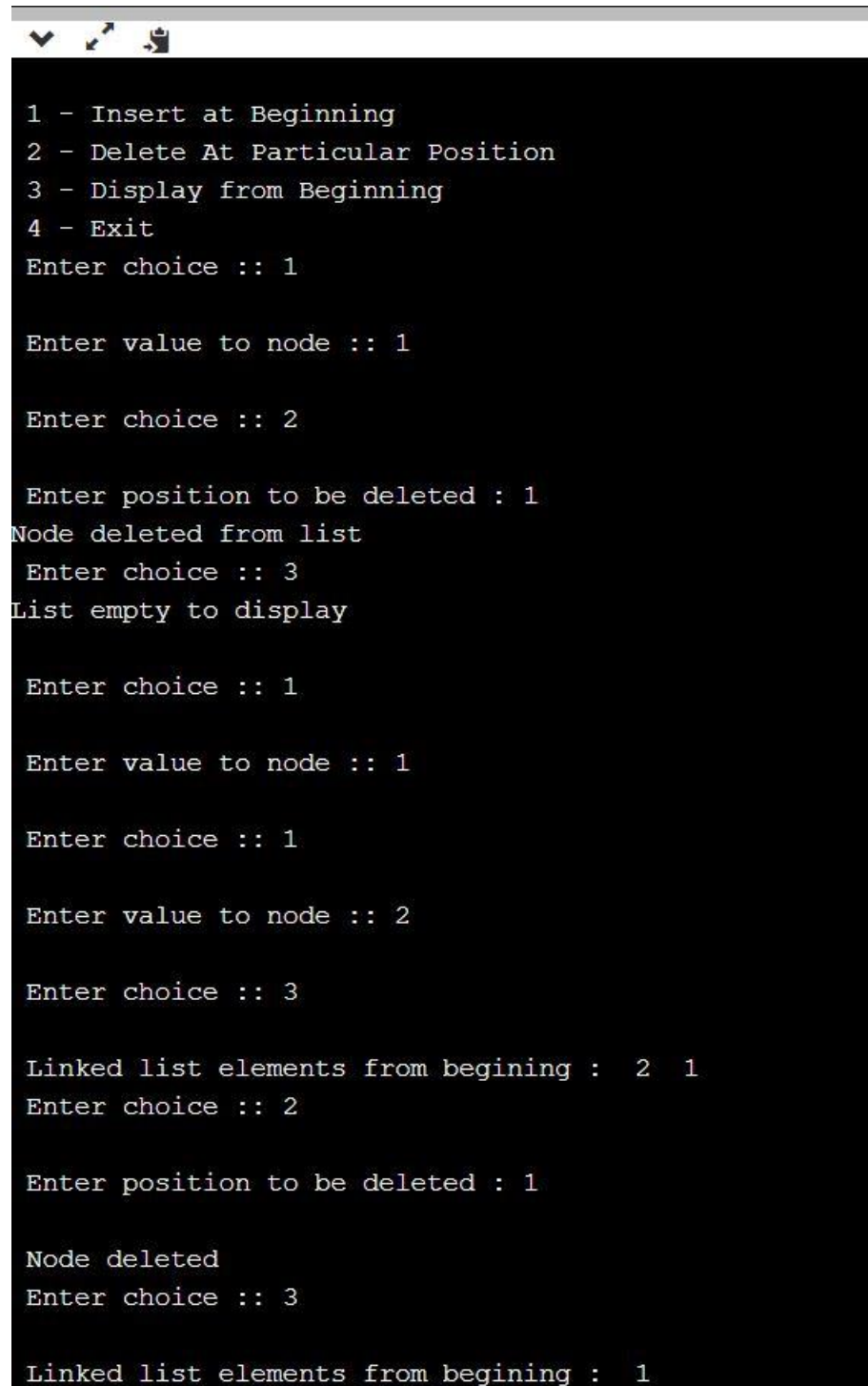
{
    temp2 = h;

    if (temp2 == NULL)
    {
        printf("List empty to display \n");
        return;
    }
    printf("\n Linked list elements from begining : ");

    while (temp2->next != NULL)
    {
        printf(" %d ", temp2->n);
    }
}

```

```
temp2 = temp2->next;
}
printf(" %d ", temp2->n);
}
```



```
1 - Insert at Beginning
2 - Delete At Particular Position
3 - Display from Beginning
4 - Exit
Enter choice :: 1

Enter value to node :: 1

Enter choice :: 2

Enter position to be deleted : 1
Node deleted from list
Enter choice :: 3
List empty to display

Enter choice :: 1

Enter value to node :: 1

Enter choice :: 1

Enter value to node :: 2

Enter choice :: 3

Linked list elements from begining : 2 1
Enter choice :: 2

Enter position to be deleted : 1

Node deleted
Enter choice :: 3

Linked list elements from begining : 1
```

