Expt no: 4

**PROGRAM: Implementation of Singly Linked List**

**Input:**

#include<stdio.h>

#include<stdlib.h>

void display();

void insert\_begin();

void insert\_end();

void insert\_pos();

void delete\_begin();

void delete\_end();

void delete\_pos();

struct node\* head = NULL;

struct node

{

int data;

struct node\* next;

};

int main()

{

int choice;

while(1)

{

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("1. display\n");

printf("2. Insert Node at beginning\n");

printf("3. Insert Node in specific position\n");

printf("4. Insert Node at end of LinkedList\n");

printf("5. Delete Node at beginning\n");

printf("6. Delete Node at end\n");

printf("7. Delete Node at position\n");

printf("8. \*\* To exit \*\*");

printf("\n Enter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1: display();

break;

case 2: insert\_begin();

break;

case 3: insert\_pos();

break;

case 4: insert\_end();

break;

case 5: delete\_begin();

break;

case 6: delete\_end();

break;

case 7: delete\_pos();

break;

case 8: exit(0);

default:printf("\n Wrong Choice");

break;

}

}

}

// prints the entire LinkedList

void display()

{

if(head==NULL)

{

printf("Linked List is Empty\n");

return;

}

printf("LinkedList: ");

struct node\* ptr = head;

// start from first node

while(ptr!=NULL)

{

printf("%d ",ptr->data);

ptr = ptr->next;

}

printf("\n");

}

// to insert node at start of LinkedList

void insert\_begin()

{

struct node\* temp;

// creating a new node

temp = (struct node\*)malloc(sizeof(struct node));

printf("Enter node data: ");

scanf("%d",&temp->data);

temp->next = NULL;

if(head==NULL)

{

head = temp;

return;

}

else

{

//point it to old head node

temp->next = head;

//point head to new first node

head = temp;

}

}

// to insert node at given position

void insert\_pos()

{

struct node\* temp;

// creating a new node

temp = (struct node\*)malloc(sizeof(struct node));

printf("Enter node data: ");

scanf("%d",&temp->data);

temp->next = NULL;

// if list empty we return

if(head==NULL)

{

head = temp;

return;

}

else

{

struct node\* prev\_ptr;

struct node\* ptr = head;

int pos;

printf("Enter position: ");

scanf("%d",&pos);

for(int i=0;i<pos;i++)

{

prev\_ptr = ptr;

ptr = ptr->next;

}

//new node pointing to node in that pos

temp->next = ptr;

//prevptr pointing to new node

prev\_ptr->next = temp;

}

}

// to insert node at end of LinkedList

void insert\_end()

{

struct node\* temp;

//creating new node

temp = (struct node\*)malloc(sizeof(struct node));

printf("Enter node data: ");

scanf("%d",&temp->data);

temp->next = NULL;

if(head==NULL)

{

//if list is empty, we return

head = temp;

return;

}

else{

struct node\* ptr = head;

while(ptr->next!=NULL)

{

ptr = ptr->next;

}

// tail node pointing to new node

ptr->next = temp;

}

}

// to delete first node of LinkedList

void delete\_begin()

{

//if List is empty we return

if(head==NULL)

{

printf("Linked List is empty | Nothing to delete \n");

return;

}

else

{

struct node\* ptr = head;

// head node pointing to second node

head = head->next;

// deleting prev head node

free(ptr);

printf("Node Deleted \n");

}

}

// to delete last node of LinkedList

void delete\_end()

{

//if List is empty we return

if(head==NULL)

{

printf("Linked List is empty | Nothing to delete \n");

return;

}

else if(head->next==NULL)

{

struct node\* ptr = head;

head = ptr->next;

free(ptr);

}

else

{

struct node\* ptr = head;

struct node\* prev\_ptr = NULL;

//traverse till last but one node

while(ptr->next!=NULL)

{

prev\_ptr = ptr;

ptr = ptr->next;

}

// next field of last but one field is made as NULL

prev\_ptr->next = NULL;

// deleting last node

free(ptr);

}

}

// to delete node at given position

void delete\_pos()

{

int pos;

printf("Enter node position to delete: ");

scanf("%d",&pos);

struct node\* ptr=head;

//we return if List is empty

if(head==NULL)

{

printf("Linked List is empty \n");

return;

}

else if(pos == 0)

{

ptr = head;

// head pointing to second node

head=ptr->next;

// deleting old first node

free(ptr);

}

else

{

struct node\* prev\_ptr;

for(int i=0;i<pos;i++)

{

prev\_ptr = ptr;

ptr = ptr->next;

}

//inserting at end of List

prev\_ptr->next = ptr->next;

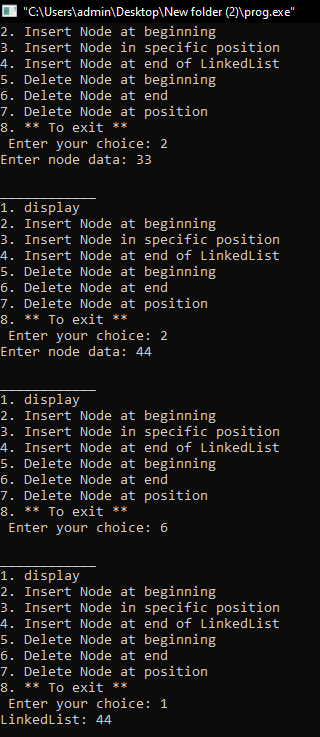
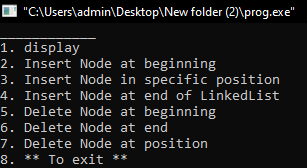
//deleting node at pos

free(ptr);

}

}

**Output:**

****