**Lab 6**

**Name : Abhinav Sanjay**

**USN : 1BM23CS009**

1) WAP to Implement Singly Linked List with following operations  
  
     a) Create a linked list.  
     b) Insertion of a node at first position, at any position and at end of list.  
    c) Display the contents of the linked list.  
  
  
2) WAP to Implement Singly Linked List with following operations  
  
   a) Create a linked list.  
   b) Deletion of first element, specified element and last element in the list.  
  c) Display the contents of the linked list.

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data; // Data for the node

struct Node \*link; // Pointer to the next node in the list

};

typedef struct Node node;

node \*start = NULL; // Start of the linked list, initially NULL

node \*new1, \*curr, \*ptr; // Global declaration for new1, curr, and ptr

// Function prototypes

void create();

void display();

void InsertStart();

void InsertPosition();

void InsertEnd();

void main() {

int ch;

while (1) {

printf("\n1. Create \n2. Display \n3. Insert at Beginning \n4. Insert at Position \n5. Insert at End \n6. Exit");

printf("\nEnter Your Choice: ");

scanf("%d", &ch);

switch (ch) {

case 1:

create();

break;

case 2:

display();

break;

case 3:

InsertStart();

break;

case 4:

InsertPosition();

break;

case 5:

InsertEnd();

break;

case 6:

exit(0);

default:

printf("Enter a Number between 1 and 6.\n");

}

}

}

// Function to create a basic linked list by appending nodes at the end

void create() {

char ch;

do {

new1 = (node\*)malloc(sizeof(node));

printf("\n enter value:\n");

scanf("%d",&new1->data);

if (start==NULL)

{

start=new1;

curr=new1;

}

else {

curr->link = new1;

curr=new1;

}

printf("Do You Want to Add an Element (Y/N)? ");

scanf(" %c", &ch);

} while (ch == 'y' || ch == 'Y');

curr->link=NULL;

}

// Insert a node at the beginning of the list

void InsertStart() {

new1 = (node\*)malloc(sizeof(node));

printf("\n enter value:\n");

scanf("%d",&new1->data);

if(start==NULL)

{

start=new1;

new1->link=NULL;

return;

}

else {

new1->link=start;

start=new1;

return;

}

}

// Insert a node at a specified position (1-based index)

void InsertPosition() {

new1 = (node\*)malloc(sizeof(node));

printf("\n enter value:\n");

scanf("%d",&new1->data);

if(start==NULL)

{

start=new1;

new1->link=NULL;

return;

}

int i=1, pos;

ptr=start;

printf("\n enter position:\n");

scanf("%d",&pos);

while (ptr!=NULL && i<pos-1)

{

ptr=ptr->link;

i++;

}

if(ptr==NULL)

{

return;

}

new1->link=ptr->link;

ptr->link=new1;

}

// Insert a node at the end of the list

void InsertEnd() {

new1 = (node\*)malloc(sizeof(node));

printf("\n enter value:\n");

scanf("%d",&new1->data);

if(start==NULL)

{

start=new1;

new1->link=NULL;

return;

}

ptr=start;

while(ptr->link !=NULL)

{

ptr=ptr->link;

}

ptr->link=new1;

new1->link=NULL;

return;

}

// Display all nodes in the linked list

void display() {

if (start == NULL) {

printf("\nLinked List is Empty.\n");

return;

}

ptr = start;

printf("\nElements in Linked List: \n");

while (ptr != NULL) {

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

ptr = ptr->link;

}

printf("\n");

}

**Output**

 