#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node\* next;

};

struct node \*head=NULL;

void traverse();

void insertAtFront();

void insertAtEnd();

void insertAtPosition();

void deleteFirst();

void deleteEnd();

void deletePosition();

int main()

{

int choice;

while (1) {

printf("\t1 For insertion at Front\n");

printf("\t2 For insertion at End\n");

printf("\t3 For deletion at Front\n");

printf("\t4 For deletion at End\n");

printf("\t5 For insertion at position\n");

printf("\t6 For deletion at position\n");

printf("\t7 To see list\n");

printf("\t8 To exit\n");

printf("Enter Choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

insertAtFront();

break;

case 2:

insertAtEnd();

break;

case 3:

deleteFirst();

break;

case 4:

deleteEnd();

break;

case 5:

insertAtPosition();

break;

case 6:

deletePosition();

break;

case 7:

traverse();

break;

case 8:exit(1);

break;

}

}

}

// Function to traverse the linked list

void traverse()

{

struct node \*temp;

// List is empty

if (head == NULL)

printf("\nList is empty\n");

// Else print the LL

temp = head;

do

{

printf("Data = %d\n", temp->data);

temp = temp->next;

}while(temp!=head);

}

// Function to insert at the front of the linked list

void insertAtFront()

{

int data;

struct node\* newnode, \*temp;

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

if (newnode == NULL) {

printf("\nUnable to allocate memory: ");

return;

}

printf("\nEnter number to be inserted : ");

scanf("%d", &data);

newnode->data = data;

if (head==NULL)

{

head=newnode;

newnode->next=head;

}

else

{

// Pointer of temp will be assigned to start

temp = head;

while (temp->next != head) {

temp = temp->next;

}

temp->next = newnode;

newnode->next=head;

head=newnode;

}

}

// Function to insert at the end of the linked list

void insertAtEnd()

{

int data;

struct node \*newnode, \*temp;

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

if(newnode == NULL) {

printf("Unable to allocate memory.");

return;

}

// Enter the number

printf("\nEnter number to be inserted : ");

scanf("%d", &data);

newnode->data = data;

if (head == NULL) {

head = newnode;

newnode->next=head;

return;

}

temp = head;

while (temp->next != head) {

temp = temp->next;

}

temp->next = newnode;

newnode->next= head;

}

void deleteFirst()

{

struct node\* temp;

if (head == NULL)

printf("\nList is empty\n");

if(head->next == head)

{

head = NULL;

free(head);

}

else {

temp = head;

while(temp->next!=head)

temp=temp->next;

temp->next = head->next;

free(head);

head=temp->next;

}

}

// Function to delete from the end of the linked list

void deleteEnd()

{

struct node\* temp,\*prev;

if (head == NULL)

printf("\nList is empty\n");

if(head->next == head)

{

head = NULL;

free(head);

}

else {

temp = head;

while(temp->next!=head)

{

prev=temp;

temp=temp->next;

}

prev->next = temp->next;

free(temp);

}

}

void insertAtPosition()

{

struct node \*temp, \*newnode;

int pos, data, i = 1;

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

if(newnode == NULL) {

printf("Unable to allocate memory.");

return;

}

// Enter the position and data

printf("\nEnter position and data :");

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

temp = head;

newnode->data = data;

//newnode->next = NULL;

while (i < pos - 1) {

temp = temp->next;

i++;

}

newnode->next = temp->next;

temp->next = newnode;

}

void deletePosition()

{

struct node \*temp, \*prev;

int i=1,pos;

// If LL is empty

if(head==NULL)

printf("\nList is empty\n");

// Otherwise

else

{

printf("\nEnter index : ");

// Position to be deleted

scanf("%d",&pos);

temp=head;

// Traverse till position

while(i<pos){

prev=temp;

temp=temp->next;

i++;

}

// Change Links

prev->next=temp->next;

// Free memory

free(temp);

}

}