#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct Node;

typedef struct Node\* node\_ptr;

typedef struct Node\* head;

typedef struct list\* List;

struct Node {

int data;

node\_ptr next;

};

struct list {

int data;

node\_ptr next;

head head;

};

List Create()

{

List new = (List)malloc(sizeof(struct list));

printf("Create Successfully!\n");

new->head = NULL;

return new;

}

void InitList(List L)

{

int num;

printf("Please input the length of the linked list: \n");

scanf("%d", &num);

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

{

int value;

printf("Please input the value of the %d-th element\n", i + 1);

scanf("%d", &value);

node\_ptr newnode = (node\_ptr)malloc(sizeof(struct Node));

newnode->data = value;

newnode->next = NULL;

if (L->head == NULL) {

L->head = newnode;

}

else {

node\_ptr current = L->head;

while (current->next != NULL) {

current = current->next;

}

current->next = newnode;

}

}

}

void Destroy(List L)

{

if (L->head == NULL) {

printf("The linked list is empty!!\n");

}

else {

node\_ptr temp, current = L->head;

while (current != NULL) {

temp = current;

current = current->next;

free(temp);

}

printf("The linked list has been destroyed!!\n");

}

}

bool isEmpty(List L)

{

return L->head == NULL;

}

int ListLength(List L)

{

node\_ptr position = L->head;

int num = 0;

while (position != NULL) {

num++;

position = position->next;

}

return num;

}

void DispList(List L)

{

node\_ptr current = L->head;

while (current != NULL)

{

printf("%d\n", current->data);

current = current->next;

}

}

node\_ptr Find(List L, int e)

{

node\_ptr current = L->head;

while (current != NULL)

{

if (current->data == e)

{

return current;

}

current = current->next;

}

return NULL;

}

node\_ptr Findpre(List L, int e)

{

node\_ptr current = L->head;

while (current->next != NULL)

{

if (current->next->data == e)

{

return current;

}

current = current->next;

}

return NULL;

}

void Insert(List L, node\_ptr p, int e)

{

node\_ptr new = (node\_ptr)malloc(sizeof(struct Node));

new->data = e;

node\_ptr position = Find(L, p);

if (position == NULL) {

new->next = NULL;

position->next = new;

}

else

{

new->next = position->next;

position->next = new;

}

}

void Delete(List L, node\_ptr p)

{

node\_ptr ptr = Find(L, p);

if (ptr == NULL)

{

printf("The element is not found!!\n");

}

else if (ptr->next == NULL)

{

printf("The element is the last one!!\n");

}

else

{

node\_ptr temp = ptr->next;

ptr->next = ptr->next->next;

free(temp);

}

}

int DeleteElem(List L, int e)

{

node\_ptr ptr = Findpre(L, e);

if (ptr == NULL)

{

printf("The element is not found!!\n");

}

else if (ptr->next == NULL)

{

printf("The element is the last one!!\n");

}

else

{

node\_ptr temp = ptr->next;

ptr->next = ptr->next->next;

free(temp);

}

}