#include <stdio.h>

#include <stdlib.h>

//////////// Node structure /////////////

struct Node {

int data;

struct Node\* next;

};

//////////// function prototype /////////////

void operation\_select(); // operation selection

int display(struct Node\* head); // show the linked list

struct Node\* insert\_at\_head(struct Node\* head, int data); //insert the given data at the begining of the list

struct Node\* insert\_at\_tail(struct Node\* head, int data); //insert the given data at the end of the list

struct Node\* delete\_at\_head(struct Node\* head); //delete the node at the head of the list

struct Node\* delete\_at\_tail(struct Node\* head); //delete the node at the end of the list

struct Node\* delete\_with\_val(struct Node\* head, int val); // find the given val and delete it from the list

void find\_element(struct Node\* head, int val); // find the given val

void count\_element(struct Node\* head); //count the length if the list

struct Node\* destroy\_list(struct Node\* head); //delete the entire list

//////////// function prototype /////////////

int main(){

struct Node\* head=NULL;

int operation, value;

do {

operation\_select();

scanf("%d",&operation);

switch (operation) {

case 1: display(head);

break;

case 2: printf("\nvalue: ");

scanf("%d", &value);

head = insert\_at\_head(head, value);

break;

case 3: printf("\nvalue: ");

scanf("%d", &value);

head = insert\_at\_tail(head, value);

break;

case 4: head = delete\_at\_head(head);

break;

case 5: head = delete\_at\_tail(head);

break;

case 6: printf("\nwith which value? ");

scanf("%d", &value);

head = delete\_with\_val(head, value);

break;

case 7: printf("\nfind which value? ");

scanf("%d", &value);

find\_element(head, value);

break;

case 8: count\_element(head);

break;

case 9: head = destroy\_list(head);

break;

case 10:

break;

default: printf("undefined input\n");

}

} while(operation!= 10);

return 0;

}

void operation\_select(){

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("select your operation (e.g., 2)\n");

printf("1- Display link list\n");

printf("2- Insert\_at\_head\n");

printf("3- Insert\_at\_tail\n");

printf("4- Delete\_at\_head\n");

printf("5- Delete\_at\_tail\n");

printf("6- Delete\_with\_val\n");

printf("7- Find\_element\n");

printf("8- Count\_element\n");

printf("9- Destroy\_list\n");

printf("10- Exit\n");

printf("Which operation? ");

}

int display(struct Node\* head){

struct Node\* curr\_node=head;

printf("\n\n");

while (curr\_node != NULL) {

printf(" %d --->", curr\_node->data);

curr\_node = curr\_node->next;

}

printf("\*NULL\*\n\n");

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

insert\_at\_head (struct Node\* head, int val)

this function adds the val into the head of the

linked list and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

insert\_at\_head (head, 42)

linked list: 42 ---> 10 --->20 ---> 30 ---> NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct Node\* insert\_at\_head(struct Node\* head, int val){

//create a link

struct Node \*link = (struct Node \*) malloc(sizeof(struct Node));

link->data = val;

//point it to old first node

link->next = head;

//point first to new first node

head = link;

free(link);

return head;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

insert\_at\_tail (struct Node\* head, int val)

this function adds the val into the tail of the

linked list and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

insert\_at\_tail (head, 42)

linked list: 10 --->20 ---> 30 ---> 42 ---> NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct Node\* insert\_at\_tail(struct Node\* head, int val){

/\*if the list was empty\*/

if(head == NULL) {

//create a link

struct Node \*link = (struct Node \*) malloc(sizeof(struct Node));

link->data = val;

link->next = NULL;

return link;

}

struct Node \*current = head;

while (current->next != NULL) {

current = current->next;

}

/\* now we can add a new variable \*/

current->next = (struct Node \*) malloc(sizeof(struct Node));

current->next->data = val;

current->next->next = NULL;

return head;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delete\_at\_head(struct Node\* head)

this function deletes the value in the head of the

linked list and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

delete\_at\_head(head)

linked list: 20 ---> 30 ---> NULL

Note: if the list is empty print out an appropriate message

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct Node\* delete\_at\_head(struct Node\* head){

if (head == NULL) {

printf("the list is empty");

return head;

}

//mark next to first link as first

head = head->next;

//return the deleted link

return head;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delete\_at\_tail(struct Node\* head)

this function deletes the value from the tail of the

linked list and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

delete\_at\_tail(head)

linked list: 10 ---> 20 ---> NULL

Note: if the list is empty print out an appropriate message

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct Node\* delete\_at\_tail(struct Node\* head){

if (head == NULL) {

printf("the list is empty");

return head;

}

/\* if there is only one item in the list, remove it \*/

if (head->next == NULL) {

free(head);

return NULL;

}

/\* get to the last node in the list \*/

struct Node \*current = head;

while (current->next->next != NULL) {

current = current->next;

}

/\* now current points to the last item of the list, so let's remove current->next \*/

free(current->next);

current->next = NULL;

return head;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

delete\_with\_val(struct Node\* head, int val)

this function deletes the node with the selected value

and returns head pointer

Example:

linked list: 10 --->20 --> 30 --> NULL

delete\_with\_val(head, 20)

linked list: 10 ---> 30 ---> NULL

Note: if the list is empty or the value is not in the

linked list print out an appropriate message

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct Node\* delete\_with\_val(struct Node\* head, int val){

//start from the first link

struct Node\* current = head;

struct Node\* previous = NULL;

//if list is empty

if(head == NULL) {

printf("Not Found! the value is not in the linked list!");

return NULL;

}

//navigate through list

while(current->data != val) {

//if it is last node

if(current->next == NULL) {

printf("Not Found! the value is not in the linked list!");

return NULL;

} else {

//store reference to current link

previous = current;

//move to next link

current = current->next;

}

}

//found a match, update the link

if(current == head) {

//change first to point to next link

head = head->next;

} else {

//bypass the current link

previous->next = current->next;

}

return head;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

find\_element(struct Node\* head, int val)

this function finds the node with the selected value

Example:

linked list: 10 --->20 --> 30 --> NULL

find\_element(head, 20)

output is : Found! the value is node number 2

find\_element(head, 42)

output is: Not Found! the value is not in the linked list!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void find\_element(struct Node\* head, int val){

//start from the first link

struct Node\* current = head;

int counter = 1;

//if list is empty

if(head == NULL) {

printf("Not Found! the value is not in the linked list!");

return;

}

//navigate through list

while(current->data != val) {

//if it is last node

if(current->next == NULL) {

printf("Not Found! the value is not in the linked list!");

return;

} else {

//go to next link

current = current->next;

counter++;

}

}

//if data found, return the current Link

printf("Found! the value is node number %i", counter);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

count\_element(struct Node\* head)

this function counts the number of node in the linked list

Example:

linked list: 10 --->20 --> 30 --> NULL

count\_element(head)

output is : 3 elements

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void count\_element(struct Node\* head){

//start from the first link

struct Node\* current = head;

int counter = 1;

//if list is empty

if(head == NULL) {

printf("0 element");

return;

}

//navigate through list

while(current->next != NULL) {

//if it is last node

current = current->next;

counter++;

}

//if data found, return the current Link

printf("%i elements", counter);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

destroy\_list(struct Node\* head)

this function removes all the nodes from the linked list

Example:

linked list: 10 --->20 --> 30 --> NULL

destroy\_list(head)

linked list: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct Node\* destroy\_list(struct Node\* head){

if(head == NULL) {

return head;

}

if(head->next != NULL){

destroy\_list(head->next);

head->next = NULL;

free(head);

}

return NULL;

}