1.Given a linked list, write a program to delete all elements with the value same as input argument ***key***, such as input linked list *a = "Head->1->2->1->3->NULL".* After calling function **Delete\_LinkList\_Node** (a, 1), the new linked list will be *"Head->2->3->NULL".*

SourceCode:

#include <stdio.h>

#include <stdlib.h>

typedef int ElemType;

typedef struct Lnode

{

ElemType data;

struct Lnode \*next;

}LNode;

LNode \*create\_LinkList(void){

int data;

LNode \*head, \*p, \*q;

head=p=(LNode \*)malloc(sizeof(LNode));

head->next=NULL;

do{

scanf("%d",& data);

q= (LNode \*)malloc(sizeof(LNode));

q->data=data;

q->next=p->next;

p->next=q;

p=q ;

}while(getchar()!='\n');

return (head);

}

void Delete\_LinkList(LNode \*L, int key){

LNode \*p=L, \*q=L->next;

while( q!=NULL && q->data!=key){

p=q;

q=q->next;

}

if(q==NULL)

printf("This key isn't in the list!\n");

else if(q->data==key){

p->next=q->next;

free(q);

}

}

void printList(LNode\* list){

printf("Head");

while(1){

printf("->");

list = list->next;

if(list == NULL){

printf("NULL");

exit(0);

}

else if(list->next == NULL) {

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

break;

}

else

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

}

}

int main(void) {

LNode \*a;

a = create\_LinkList();

printList(a);

int elem=6;

Delete\_LinkList(a, elem);

printList(a);

return 0;

}

Output:

1 2 3 4

Head->1->2->3->4->NULL

This key isn't in the list!

Head->1->2->3->4->NULL

1 2 3 6 8

Head->1->2->3->6->8->NULL

Head->1->2->3->8->NULL

2. From a linked list, generate a program to delete all elements with the same value, such as input linked list *a = "Head->1->2->1->3->2->NULL".* After calling function **Delete\_Node\_value**(a), the new linked list will be *"Head->3->NULL".*

*SourceCode:*

#include <stdio.h>

#include <stdlib.h>

typedef int ElemType;

typedef struct Lnode

{

ElemType data;

struct Lnode \*next;

}LNode;

LNode \*create\_LinkList(void){

int data;

LNode \*head, \*p, \*q;

head=p=(LNode \*)malloc(sizeof(LNode));

head->next=NULL;

do{

scanf("%d",& data);

q= (LNode \*)malloc(sizeof(LNode));

q->data=data;

q->next=p->next;

p->next=q;

p=q ;

}while(getchar()!='\n');

return (head);

}

void Delete\_Node\_value(LNode \*L){

//1 count the length

LNode\* cur = L->next;

int len = 0;

while (cur != NULL) {

len++;

cur = cur->next;

}

//2 creat array of values

cur = L->next;

int i = 0;

int values[len];

while (cur != NULL) {

values[i] = cur->data;

i++;

cur = cur->next;

}

//3 decide which ones to keep and which ones to delete

int keep[len];

for (i = 0; i < len; i++) {

keep[i] = 1;

}

for (i = 0; i < len; i++) {

int j;

for (j = i + 1; j < len; j++) {

if (values[i] == values[j]) {

keep[i] = 0;

keep[j] = 0;

}

}

}

//4 rebuild linked list

i = 0;

cur = L->next;

LNode\* prev = L;

while (cur != NULL) {

LNode\* next = cur->next;

if (keep[i] == 0) {

prev->next = cur->next;

free(cur);

} else {

prev = cur;

}

i++;

cur = next;

}

}

void printList(LNode\* list){

printf("Head");

while(1){

printf("->");

list = list->next;

if(list == NULL){

printf("NULL");

exit(0);

}

else if(list->next == NULL) {

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

break;

}

else

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

}

}

int main(void) {

LNode \*a;

a = create\_LinkList();

printList(a);

Delete\_Node\_value(a);

printList(a);

return 0;

}

*Ouput:*

*1 1 1 1 1 2 2 2 3 2 2 2 4*

*Head->1->1->1->1->1->2->2->2->3->2->2->2->4->NULL*

*Head->3->4->NULL*

3. In a linked list, create a program to delete all duplicated elements, such as input linked list *a = "Head->1->2->1->3->2->NULL"*. After calling function **Delete\_Dupl\_Node(a)**, the new linked list will be *"Head->1->2->3->NULL".*

SourceCode:

#include <stdio.h>

#include <stdlib.h>

typedef int ElemType;

typedef struct Lnode

{

ElemType data;

struct Lnode \*next;

}LNode;

LNode \*create\_LinkList(void){

int data;

LNode \*head, \*p, \*q;

head=p=(LNode \*)malloc(sizeof(LNode));

head->next=NULL;

do{

scanf("%d",& data);

q= (LNode \*)malloc(sizeof(LNode));

q->data=data;

q->next=p->next;

p->next=q;

p=q ;

}while(getchar()!='\n');

return (head);

}

void Delete\_Dupl\_Node(LNode \*L) {

LNode\* cur = L->next;

while (cur != NULL) {

LNode\* prev = cur;

LNode\* check = cur->next;

while (check != NULL) {

LNode\* next = check->next;

if (check->data == cur->data) {

prev->next = check->next;

free(check);

} else {

prev = check;

}

check = next;

}

cur = cur->next;

}

}

void printList(LNode\* list){

printf("Head");

while(1){

printf("->");

list = list->next;

if(list->next == NULL) {

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

break;

}

else

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

}

}

int main(void) {

LNode \*a;

a = create\_LinkList();

printList(a);

Delete\_Dupl\_Node(a);

printList(a);

return 0;

}

Output:

1 2 1 2

Head->1->2->1->2->NULL

Head->1->2->NULL

1 2 3 3 2 1

Head->1->2->3->3->2->1->NULL

Head->1->2->3->NULL

1 2 2 2 2 2

Head->1->2->2->2->2->2->NULL

Head->1->2->NULL

1 2 3 1 1

Head->1->2->3->1->1->NULL

Head->1->2->3->NULL

1 1

Head->1->1->NULL

Head->1->NULL

4. Write a program to reverse given linked list, like input linked list

*a = "Head->1->2->3->NULL"*. Through calling function **Reverse\_Node**(a), the new linked list will be *"Head->3->2->1->NULL"*.

SourceCode:

#include <stdio.h>

#include <stdlib.h>

typedef int ElemType;

typedef struct Lnode

{

ElemType data;

struct Lnode \*next;

}LNode;

LNode \*create\_LinkList(void){

int data;

LNode \*head, \*p, \*q;

head=p=(LNode \*)malloc(sizeof(LNode));

head->next=NULL;

do{

scanf("%d",& data);

q= (LNode \*)malloc(sizeof(LNode));

q->data=data;

q->next=p->next;

p->next=q;

p=q ;

}while(getchar()!='\n');

return (head);

}

void reverseList(LNode \*L) {

LNode\* head = L;

LNode\* cur = head->next;

LNode\* prev = NULL;

while(cur != NULL) {

LNode\* tmp = cur->next;

cur->next = prev;

prev = cur;

cur = tmp;

}

head->next = prev;

}

void printList(LNode\* list){

printf("Head");

while(1){

printf("->");

list = list->next;

if(list->next == NULL) {

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

break;

}

else

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

}

}

int main(void) {

LNode \*a;

a = create\_LinkList();

printList(a);

reverseList(a);

printList(a);

return 0;

}

Output:

1 2 3 4 5

Head->1->2->3->4->5->NULL

Head->5->4->3->2->1->NULL

5. From given a linked list with integer value nodes, calculate average value, for instance, input linked list *a = "Head->1->2->3->NULL".* By calling function **Average\_List**(a), average value is (1+2+3) / 3=2.0 (floating data type).

SourceCode:

#include <stdio.h>

#include <stdlib.h>

typedef int ElemType;

typedef struct Lnode

{

ElemType data;

struct Lnode \*next;

}LNode;

LNode \*create\_LinkList(void){

int data;

LNode \*head, \*p, \*q;

head=p=(LNode \*)malloc(sizeof(LNode));

head->next=NULL;

do{

scanf("%d",& data);

q= (LNode \*)malloc(sizeof(LNode));

q->data=data;

q->next=p->next;

p->next=q;

p=q ;

}while(getchar()!='\n');

return (head);

}

float averageList(LNode \*L) {

LNode\* cur = L->next;

int sum=0;

int count=0;

float average=0.0;

while (cur != NULL){

sum += cur->data;

count++;

cur = cur->next;

}

average=sum \* 1.0f/count;

return average;

}

void printList(LNode\* list){

printf("Head");

while(1){

printf("->");

list = list->next;

if(list->next == NULL) {

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

break;

}

else

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

}

}

int main(void) {

LNode \*a;

a = create\_LinkList();

printList(a);

float result = averageList(a);

printf("%f\n", result);

return 0;

}

Output:

1 2 3 4 5

Head->1->2->3->4->5->NULL

3.000000