DSA Assignment-04

Name: Palak Parhawk

Roll No.: 21052087



Ans:



Ans:

#include <stdio.h>

#include <stdlib.h>

/\* A structure of linked list node \*/

struct node {

int data;

struct node \*next;

} \*head;

void initialize(){

head = NULL;

}

/\*

Given a Inserts a node in front of a singly linked list.

\*/

void insert(int num) {

/\* Create a new Linked List node \*/

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

newNode->data = num;

/\* Next pointer of new node will point to head node of linked list \*/

newNode->next = head;

/\* make new node as new head of linked list \*/

head = newNode;

printf("Inserted Element : %d\n", num);

}

/\*

Prints Linked List in reverse order without reversing it.

\*/

void reverseLLPrint(struct node \*head) {

if (head != NULL) {

reverseLLPrint(head->next);

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

}

}

/\*

Prints a linked list from head node till tail node

\*/

void printLinkedList(struct node \*nodePtr) {

while (nodePtr != NULL) {

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

nodePtr = nodePtr->next;

if(nodePtr != NULL)

printf("-->");

else

printf("-->NULL");

}

}

int main() {

initialize();

/\* Creating a linked List\*/

insert(8);

insert(3);

insert(2);

insert(7);

insert(9);

printf("\nLinked List\n");

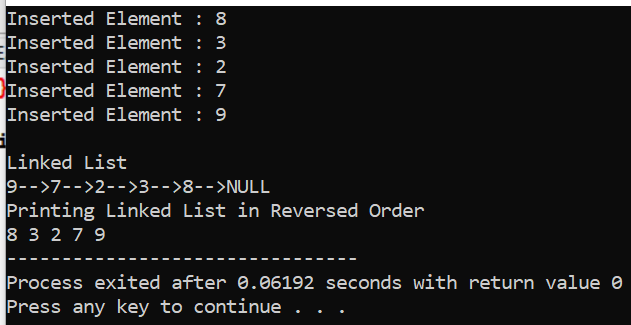
printLinkedList(head);

printf("\nPrinting Linked List in Reversed Order\n");

reverseLLPrint(head);

return 0;

}



1. WAP to print mth node from the last of a linked list of n nodes.

Ans:

#include<stdio.h>

#include<stdlib.h>

//structure of a node

struct node{

int data;

struct node \*next;

}\*head,\*temp;

int count=0;

//function for inserting nodes into a list

void insert(int val){

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

newnode->data = val;

newnode->next = NULL;

if(head == NULL){

head = newnode;

temp = head;

count++;

} else {

temp->next=newnode;

temp=temp->next;

count++;

}

}

//function for displaying a list

void display(){

if(head==NULL)

printf("no node ");

else {

temp=head;

while(temp!=NULL) {

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

temp=temp->next;

}

}

}

//function for finding 3rd node from the last of a linked list

void last(int n){

int i;

temp=head;

for(i=0;i<count-n;i++){

temp=temp->next;

}

printf("\n%drd node from the end of linked list is : %d" ,n,temp->data);

}

int main(){

//creating list

struct node\* head = NULL;

int n=3;

//inserting elements into a list

insert(1);

insert(2);

insert(3);

insert(4);

insert(5);

insert(6);

//displaying the list

printf("\nlinked list is : ");

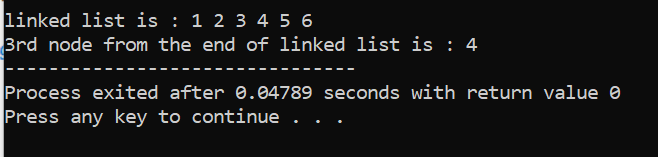
display();

//calling function for finding nth element in a list from last

last(n);

return 0;

}



HA



Ans:



Ans:

// C program to count occurrences in a linked list

#include <stdio.h>

#include <stdlib.h>

/\* Link list node \*/

struct Node {

int data;

struct Node\* next;

};

/\* Given a reference (pointer to pointer) to the head

of a list and an int, push a new node on the front

of the list. \*/

void push(struct Node\*\* head\_ref, int new\_data)

{

/\* allocate node \*/

struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));

/\* put in the data \*/

new\_node->data = new\_data;

/\* link the old list off the new node \*/

new\_node->next = (\*head\_ref);

/\* move the head to point to the new node \*/

(\*head\_ref) = new\_node;

}

/\* Counts the no. of occurrences of a node

(search\_for) in a linked list (head)\*/

int count(struct Node\* head, int search\_for)

{

struct Node\* current = head;

int count = 0;

while (current != NULL) {

if (current->data == search\_for)

count++;

current = current->next;

}

return count;

}

/\* Driver program to test count function\*/

int main()

{

/\* Start with the empty list \*/

struct Node\* head = NULL;

/\* Use push() to construct below list

1->2->1->3->1 \*/

push(&head, 1);

push(&head, 3);

push(&head, 1);

push(&head, 2);

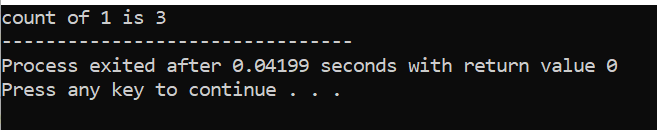
push(&head, 1);

/\* Check the count function \*/

printf("count of 1 is %d", count(head, 1));

return 0;

}





Ans:

/\*

\* C Program to Reverse only First N Elements of a Linked List

\*/

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \*next;

};

void create(struct node \*\*);

void reverse(struct node \*\*, int);

void release(struct node \*\*);

void display(struct node \*);

int main()

{

struct node \*p = NULL;

int n;

printf("Enter data into the list\n");

create(&p);

printf("Displaying the nodes in the list:\n");

display(p);

printf("Enter the number N to reverse first N node: ");

scanf("%d", &n);

printf("Reversing the list...\n");

if (n > 1)

{

reverse(&p, n - 2);

}

printf("Displaying the reversed list:\n");

display(p);

release(&p);

return 0;

}

void reverse(struct node \*\*head, int n)

{

struct node \*p, \*q, \*r, \*rear;

p = q = r = \*head;

if (n == 0)

{

q = q->next;

p->next = q->next;

q->next = p;

\*head = q;

}

else

{

p = p->next->next;

q = q->next;

r->next = NULL;

rear = r;

q->next = r;

while (n > 0 && p != NULL)

{

r = q;

q = p;

p = p->next;

q->next = r;

n--;

}

\*head = q;

rear->next = p;

}

}

void create(struct node \*\*head)

{

int c, ch;

struct node \*temp, \*rear;

do

{

printf("Enter number: ");

scanf("%d", &c);

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

temp->num = c;

temp->next = NULL;

if (\*head == NULL)

{

\*head = temp;

}

else

{

rear->next = temp;

}

rear = temp;

printf("Do you wish to continue [1/0]: ");

scanf("%d", &ch);

} while (ch != 0);

printf("\n");

}

void display(struct node \*p)

{

while (p != NULL)

{

printf("%d\t", p->num);

p = p->next;

}

printf("\n");

}

void release(struct node \*\*head)

{

struct node \*temp = \*head;

\*head = (\*head)->next;

while ((\*head) != NULL)

{

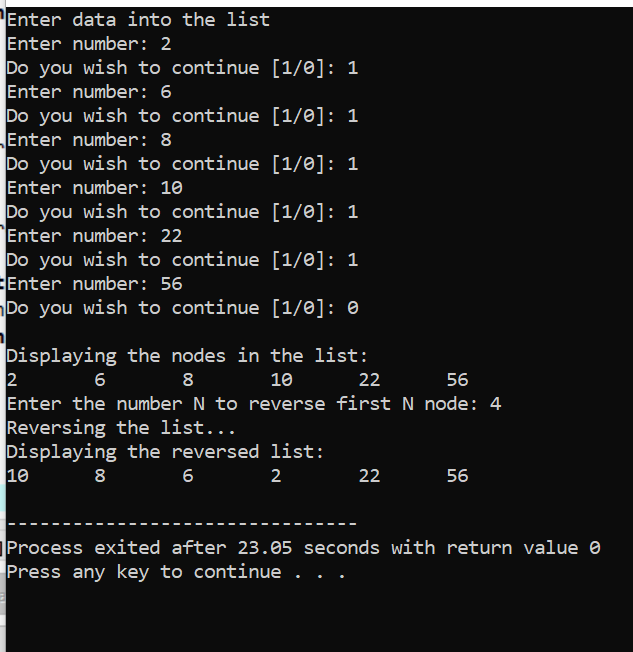
free(temp);

temp = \*head;

(\*head) = (\*head)->next;

}

}





Ans:

/\* C Program to remove duplicates in an unsorted

linked list \*/

#include<stdio.h>

#include<stdlib.h>

/\* A linked list node \*/

typedef struct Node {

int data;

struct Node\* next;

} Node;

// Utility function to create a new Node

Node\* newNode(int data)

{

Node\* temp = (Node\*)malloc(sizeof(Node));

temp->data = data;

temp->next = NULL;

return temp;

}

/\* Function to remove duplicates from a

unsorted linked list \*/

void removeDuplicates(Node\* start)

{

Node \*ptr1, \*ptr2, \*dup;

ptr1 = start;

/\* Pick elements one by one \*/

while (ptr1 != NULL && ptr1->next != NULL) {

ptr2 = ptr1;

/\* Compare the picked element with rest

of the elements \*/

while (ptr2->next != NULL) {

/\* If duplicate then delete it \*/

if (ptr1->data == ptr2->next->data) {

/\* sequence of steps is important here \*/

dup = ptr2->next;

ptr2->next = ptr2->next->next;

free(dup);

}

else /\* This is tricky \*/

ptr2 = ptr2->next;

}

ptr1 = ptr1->next;

}

}

/\* Function to print nodes in a given linked list \*/

void printList(struct Node\* node)

{

while (node != NULL) {

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

node = node->next;

}

}

/\* Driver program to test above function \*/

int main()

{

/\* The constructed linked list is:

10->12->11->11->12->11->10\*/

struct Node\* start = newNode(10);

start->next = newNode(12);

start->next->next = newNode(11);

start->next->next->next = newNode(11);

start->next->next->next->next = newNode(12);

start->next->next->next->next->next = newNode(11);

start->next->next->next->next->next->next = newNode(10);

printf("Linked list before removing duplicates ");

printList(start);

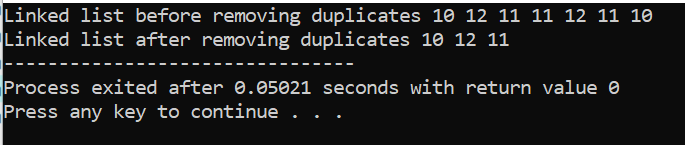
removeDuplicates(start);

printf("\nLinked list after removing duplicates ");

printList(start);

return 0;

}





Ans:

/\* Program to insert in a sorted list \*/

#include <stdio.h>

#include <stdlib.h>

/\* Link list node \*/

struct Node {

int data;

struct Node\* next;

};

/\* function to insert a new\_node

in a list. Note that this

function expects a pointer

to head\_ref as this can modify the

head of the input linked

list (similar to push())\*/

void sortedInsert(struct Node\*\* head\_ref,

struct Node\* new\_node)

{

struct Node\* current;

/\* Special case for the head end \*/

if (\*head\_ref == NULL

|| (\*head\_ref)->data

>= new\_node->data) {

new\_node->next = \*head\_ref;

\*head\_ref = new\_node;

}

else {

/\* Locate the node before

the point of insertion \*/

current = \*head\_ref;

while (current->next != NULL

&& current->next->data < new\_node->data) {

current = current->next;

}

new\_node->next = current->next;

current->next = new\_node;

}

}

/\* BELOW FUNCTIONS ARE JUST UTILITY TO TEST sortedInsert \*/

/\* A utility function to create a new node \*/

struct Node\* newNode(int new\_data)

{

/\* allocate node \*/

struct Node\* new\_node

= (struct Node\*)malloc(

sizeof(struct Node));

/\* put in the data \*/

new\_node->data = new\_data;

new\_node->next = NULL;

return new\_node;

}

/\* Function to print linked list \*/

void printList(struct Node\* head)

{

struct Node\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

}

/\* Driver program to test count function\*/

int main()

{

/\* Start with the empty list \*/

struct Node\* head = NULL;

struct Node\* new\_node = newNode(5);

sortedInsert(&head, new\_node);

new\_node = newNode(10);

sortedInsert(&head, new\_node);

new\_node = newNode(7);

sortedInsert(&head, new\_node);

new\_node = newNode(3);

sortedInsert(&head, new\_node);

new\_node = newNode(1);

sortedInsert(&head, new\_node);

new\_node = newNode(9);

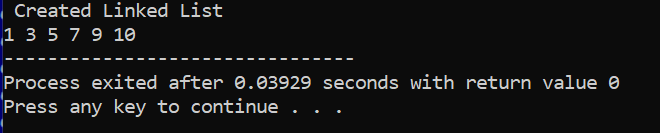
sortedInsert(&head, new\_node);

printf("\n Created Linked List\n");

printList(head);

return 0;

}





Ans:

/\*

\* C Program to Find Number of Occurences of All Elements in a Linked List

\*/

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \*next;

};

struct node\_occur

{

int num;

int times;

struct node\_occur \*next;

};

void create(struct node \*\*);

void occur(struct node \*, struct node\_occur \*\*);

void release(struct node \*\*);

void release\_2(struct node\_occur \*\*);

void display(struct node \*);

void disp\_occur(struct node\_occur \*);

int main()

{

struct node \*p = NULL;

struct node\_occur \*head = NULL;

int n;

printf("Enter data into the list\n");

create(&p);

printf("Displaying the occurence of each node in the list:\n");

display(p);

occur(p, &head);

disp\_occur(head);

release(&p);

release\_2(&head);

return 0;

}

void occur(struct node \*head, struct node\_occur \*\*result)

{

struct node \*p;

struct node\_occur \*temp, \*prev;

p = head;

while (p != NULL)

{

temp = \*result;

while (temp != NULL && temp->num != p->num)

{

prev = temp;

temp = temp->next;

}

if (temp == NULL)

{

temp = (struct node\_occur \*)malloc(sizeof(struct node\_occur));

temp->num = p->num;

temp->times = 1;

temp->next = NULL;

if (\*result != NULL)

{

prev->next = temp;

}

else

{

\*result = temp;

}

}

else

{

temp->times += 1;

}

p = p->next;

}

}

void create(struct node \*\*head)

{

int c, ch;

struct node \*temp, \*rear;

do

{

printf("Enter number: ");

scanf("%d", &c);

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

temp->num = c;

temp->next = NULL;

if (\*head == NULL)

{

\*head = temp;

}

else

{

rear->next = temp;

}

rear = temp;

printf("Do you wish to continue [1/0]: ");

scanf("%d", &ch);

} while (ch != 0);

printf("\n");

}

void display(struct node \*p)

{

while (p != NULL)

{

printf("%d\t", p->num);

p = p->next;

}

printf("\n");

}

void disp\_occur(struct node\_occur \*p)

{

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

while (p != NULL)

{

printf(" %d\t\t%d\n", p->num, p->times);

p = p->next;

}

}

void release(struct node \*\*head)

{

struct node \*temp = \*head;

\*head = (\*head)->next;

while ((\*head) != NULL)

{

free(temp);

temp = \*head;

(\*head) = (\*head)->next;

}

}

void release\_2(struct node\_occur \*\*head)

{

struct node\_occur \*temp = \*head;

\*head = (\*head)->next;

while ((\*head) != NULL)

{

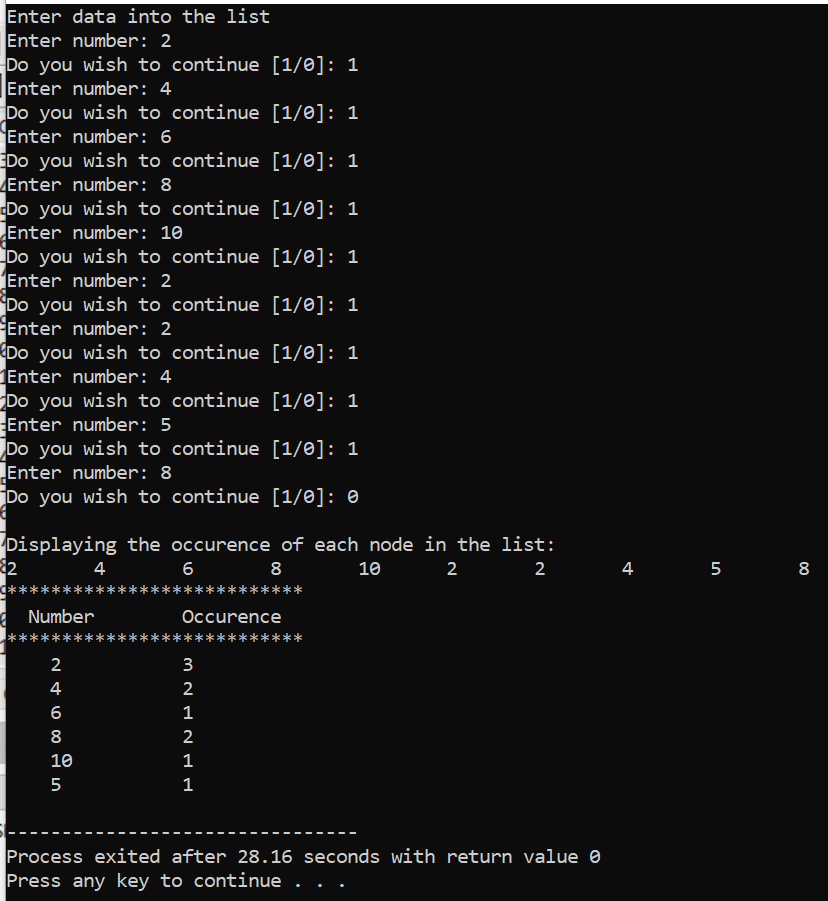
free(temp);

temp = \*head;

(\*head) = (\*head)->next;

}

}





Ans:

/\*

\* C Program to Modify the Linked List such that All Even Numbers

\* appear before all the Odd Numbers in the Modified Linked List

\*/

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \*next;

};

void create(struct node \*\*);

void generate\_evenodd(struct node \*, struct node\*\*);

void release(struct node \*\*);

void display(struct node \*);

int main()

{

struct node \*p = NULL, \*q = NULL;

int key, result;

printf("Enter data into the list\n");

create(&p);

printf("Displaying the nodes in the list:\n");

display(p);

generate\_evenodd(p, &q);

printf("Displaying the list with even and then odd:\n");

display(q);

release(&p);

return 0;

}

void generate\_evenodd(struct node \*list, struct node \*\*head)

{

struct node \*even = NULL, \*odd = NULL, \*temp;

struct node \*reven, \*rodd;

while (list != NULL)

{

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

temp->num = list->num;

temp->next = NULL;

if (list->num % 2 == 0)

{

if (even == NULL)

{

even = temp;

}

else

{

reven->next = temp;

}

reven = temp;

}

else

{

if (odd == NULL)

{

odd = temp;

}

else

{

rodd->next = temp;

}

rodd = temp;

}

list = list->next;

}

reven->next = odd;

\*head = even;

}

void create(struct node \*\*head)

{

int c, ch;

struct node \*temp, \*rear;

do

{

printf("Enter number: ");

scanf("%d", &c);

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

temp->num = c;

temp->next = NULL;

if (\*head == NULL)

{

\*head = temp;

}

else

{

rear->next = temp;

}

rear = temp;

printf("Do you wish to continue [1/0]: ");

scanf("%d", &ch);

} while (ch != 0);

printf("\n");

}

void display(struct node \*p)

{

while (p != NULL)

{

printf("%d\t", p->num);

p = p->next;

}

printf("\n");

}

void release(struct node \*\*head)

{

struct node \*temp = \*head;

\*head = (\*head)->next;

while ((\*head) != NULL)

{

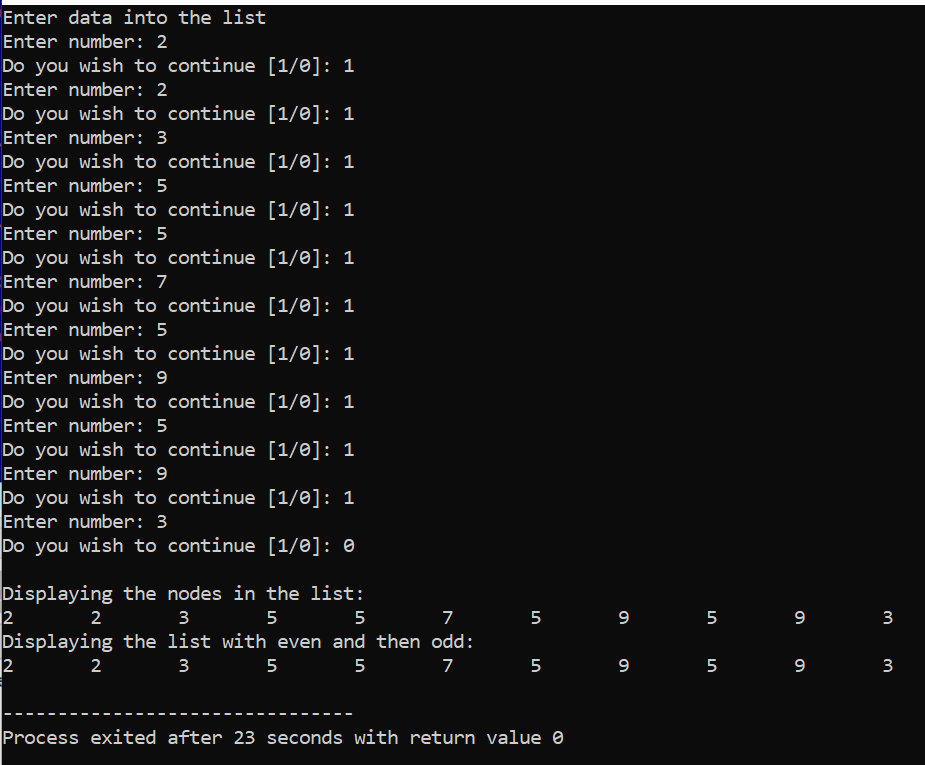
free(temp);

temp = \*head;

(\*head) = (\*head)->next;

}

}





Ans:

/\*

\* C Program to Check whether a Singly Linked List is a Palindrome

\*/

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \*next;

};

int create(struct node \*\*);

int palin\_check (struct node \*, int);

void release(struct node \*\*);

int main()

{

struct node \*p = NULL;

int result, count;

printf("Enter data into the list\n");

count = create(&p);

result = palin\_check(p, count);

if (result == 1)

{

printf("The linked list is a palindrome.\n");

}

else

{

printf("The linked list is not a palindrome.\n");

}

release (&p);

return 0;

}

int palin\_check (struct node \*p, int count)

{

int i = 0, j;

struct node \*front, \*rear;

while (i != count / 2)

{

front = rear = p;

for (j = 0; j < i; j++)

{

front = front->next;

}

for (j = 0; j < count - (i + 1); j++)

{

rear = rear->next;

}

if (front->num != rear->num)

{

return 0;

}

else

{

i++;

}

}

return 1;

}

int create (struct node \*\*head)

{

int c, ch, count = 0;

struct node \*temp;

do

{

printf("Enter number: ");

scanf("%d", &c);

count++;

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

temp->num = c;

temp->next = \*head;

\*head = temp;

printf("Do you wish to continue [1/0]: ");

scanf("%d", &ch);

}while (ch != 0);

printf("\n");

return count;

}

void release (struct node \*\*head)

{

struct node \*temp = \*head;

while ((\*head) != NULL)

{

(\*head) = (\*head)->next;

free(temp);

temp = \*head;

}

}

