**Coding for First 10 Questions**

**Write a program to create a singly linked list and display its elements.**

**Insert a node at the beginning, middle, and end of a singly linked list.**

**Delete a node from the beginning, middle, and end of a singly linked list.**

**Count the number of nodes in a linked list.**

**Search for a given value in a linked list.**

**Reverse a singly linked list.**

**Merge two sorted linked lists into a single sorted list.**

**Find the nth node from the end of a linked list.**

**Detect a loop in a linked list and remove it.**

**Check if a linked list is a palindrome.**

#include <stdio.h>

#include <stdlib.h>

struct Node{

int data;

struct Node \*next;

};

struct Node\* createNode(int data){

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

newnode->data = data;

newnode->next =NULL;

return newnode;

}

void insertAtEnd(struct Node \*\*head,int data){

struct Node \*newnode = createNode(data);

printf("A\n");

if(\*head==NULL){

\*head = newnode;

return;

}

struct Node \*temp = \*head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=newnode;

printf("End\n");

}

void insertAtBegin(struct Node \*\*head,int data){

struct Node \*newnode = createNode(data);

if(\*head==NULL){

\*head = newnode;

return;

}

//struct Node \*temp = \*head;

newnode->next = \*head;

\*head =newnode;

}

//2 5 10 15

// 12

void insertAtPos(struct Node \*\*head,int data,int pos){

struct Node \*newnode = createNode(data);

struct Node \*temp = \*head;

if(pos==1){

insertAtBegin(head,data);

return;

}

for(int i=1;i<pos-1;i++){

temp=temp->next;

}

if(temp==NULL){

printf("Position Out of Range");

return;

}

newnode->next = temp->next;

temp->next = newnode;

}

void deleteAtEnd(struct Node \*\*head){

struct Node \*temp = \*head;

if((\*head)->next==NULL){

free(\*head);

\*head=NULL;

return;

}

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

temp=temp->next;

}

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

temp->next = NULL;

}

void deleteAtBegin(struct Node \*\*head){

struct Node \*temp = \*head;

if(temp->next==NULL){

free(head);

\*head=NULL;

return;

}

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

free(temp);

}

void deleteAtpos(struct Node \*\*head,int pos){

struct Node \*temp = \*head;

for(int i=1;i<pos-1;i++){

temp = temp->next;

}

temp->next=temp->next->next;

struct Node \*del = temp->next;

free(del);

}

int search(struct Node \*head,int value){

struct Node \*temp=head;

while(temp!=NULL){

if(temp->data==value){

return 1;

}

temp=temp->next;

}

return 0;

}

void reverse(struct Node \*\*head){

struct Node \*temp = \*head;

struct Node \*prev = NULL;

struct Node \*next = NULL;

while(temp!=NULL){

next = temp->next;

temp->next = prev;

prev = temp;

temp = next;

}

\*head=prev;

}

struct Node\* mergelist(struct Node \*h1,struct Node \*h2){

struct Node dummy;

dummy.next=NULL;

struct Node \*tail = &dummy;

while(h1!=NULL && h2!=NULL){

if(h1->data<=h2->data){

tail->next=h1;

h1=h1->next;

}

else{

tail->next=h2;

h2 = h2->next;

}

tail = tail->next;

}

if(h1!=NULL)

tail->next = h1;

else if(h2!=NULL)

tail->next = h2;

return dummy.next;

}

//2 1 3

void bubbleSort(struct Node \*head){

struct Node \*current;

int flag;

struct Node \*last = NULL;

do{

flag=0;

current = head;

while(current->next!=last){

if(current->data > current->next->data){

int temp = current->next->data;//1

current->next->data = current->data;

current->data = temp;

flag=1;

}

current=current->next;

}

last = current;

}while(flag);

}

//2 5 10 15 12

void findnth(struct Node \*head,int n){

struct Node \*refptr=head,\*mainptr=head;

for(int i=0;i<n;i++){

if(refptr==NULL){

return;

}

refptr=refptr->next;

}

while(refptr!=NULL){

mainptr=mainptr->next;//15

refptr=refptr->next;//NULL

}

printf("\nNth Node is %d\n",mainptr->data);

}

void removal(struct Node \*head){

struct Node \*fast=head,\*slow=head;

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

fast = fast->next->next;

slow=slow->next;

if(fast==slow){

printf("Loop Detected");

break;

}

}

if(fast==NULL || fast->next==NULL){

printf("No Loop Formed");

return;

}

struct Node \*prev=NULL;

int count=0;

slow=head;

while(slow!=fast){

prev = fast;

slow=slow->next;

fast = fast->next;

count++;

}

prev->next = NULL;

printf("\n%d\n",count);

}

int palindrome(struct Node \*head){

struct Node \*current = head;

int arr[5];

int size=0;

while(current!=NULL){

arr[size++]=current->data;

current=current->next;

}

int i=0,j=size-1;

while(i<j){

if(arr[i]!=arr[j]){

return 0;

}

i++;

j--;

}

return 1;

}

void display(struct Node \*head){

struct Node \*temp = head;

int count=0;

while(temp!=NULL){

count++;

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

temp=temp->next;

}

printf("\nCount of the Nodes %d",count);

}

int main()

{

struct Node \*head=NULL;

insertAtEnd(&head,3);

insertAtEnd(&head,2);

insertAtBegin(&head,2);

insertAtBegin(&head,1);

insertAtEnd(&head,1);

// display(head);

// printf("\n");

// findnth(head,3); 2 5 10 15 12

//head->next->next->next->next->next = head->next->next;

//removal(head);

//display(&head);

//printf("\nDeletion Starts\n");

//deleteAtEnd(&head);

// deleteAtBegin(&head);

// deleteAtpos(&head,2);

// //display(&head);

// int found = search(head,6);

// if(found)

// printf("\nIts there");

// else

// printf("\nIts not there");

//reverse(&head);

// struct Node \*head2 = NULL;

// insertAtEnd(&head2,1);

// insertAtEnd(&head2,4);

// insertAtEnd(&head2,3);

//display(head2);

// bubbleSort(head);

// bubbleSort(head2);

// struct Node \*res = mergelist(head,head2);

//printf("Before Calling Find Nth: ");

//display(head);

// findnth(head,2);

//display(res);

display(head);

if(palindrome(head))

printf("Its a Palindrome");

else

printf("Not a Palindrome");

return 0;

}

**Coding for Next Three Questions**

**Remove duplicate nodes from an unsorted linked list.**

**Find the intersection point of two linked lists.**

**Rotate a linked list to the left or right by k positions.**#include <stdio.h>

#include <stdlib.h>

struct Node{

int data;

struct Node \*next;

};

struct Node\* createNode(int data){

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

newnode->data = data;

newnode->next =NULL;

return newnode;

}

void insertAtEnd(struct Node \*\*head,int data){

struct Node \*newnode = createNode(data);

printf("A\n");

if(\*head==NULL){

\*head = newnode;

return;

}

struct Node \*temp = \*head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=newnode;

printf("End\n");

}

void duplicates(struct Node \*head){

struct Node \*current = head,\*temp,\*prev;

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

prev = current;

while(prev->next!=NULL){

if(current->data==prev->next->data){

temp = prev->next;

prev->next = prev->next->next;

free(temp);

}

else

prev = prev->next;

}

current=current->next;

}

}

int findlength(struct Node \*head){

int len = 0;

struct Node \*temp = head;

while(temp!=NULL){

len++;

temp=temp->next;

}

return len;

}

struct Node\* findinter(struct Node \*head,struct Node \*head2){

int l1 = findlength(head);

int l2 = findlength(head2);

int diff = abs(l1-l2);

struct Node \*longer = (l1>l2)?head:head2;

struct Node \*shorter = (l1>l2)?head2:head;

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

longer=longer->next;

while(longer!=NULL && shorter!=NULL){

if(longer==shorter)

return longer;

longer = longer->next;

shorter=shorter->next;

}

};

struct Node\* leftrotate(struct Node \*head,int n){

struct Node \*current = head;

int len =findlength(head);

int k = n%len;

if(k==0)

return head;

for(int i=1;i<k;i++)

current = current->next;

struct Node \*newhead = current->next;

current->next = NULL;

struct Node \*temp = newhead;

while(temp->next!=NULL)

temp = temp->next;

temp->next = head;

return newhead;

}

struct Node\* rightrotate(struct Node \*head,int k){

if(head==NULL || k==0)

return head;

int len = findlength(head);

k = k%len;

return leftrotate(head,len-k);

}

void display(struct Node \*head){

struct Node \*temp = head;

int count=0;

while(temp!=NULL){

count++;

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

temp=temp->next;

}

printf("\nCount of the Nodes %d",count);

}

int main()

{

struct Node \*head=NULL;

insertAtEnd(&head,1);

insertAtEnd(&head,2);

insertAtEnd(&head,3);

insertAtEnd(&head,4);

insertAtEnd(&head,5);

//insertAtBegin(&head,2);

/\*struct Node \*head2 = NULL;

insertAtEnd(&head2,1);

insertAtEnd(&head2,4);

//duplicates(head);

struct Node \*common = NULL;

insertAtEnd(&common,5);

insertAtEnd(&common,6);

insertAtEnd(&common,7);

struct Node \*temp = head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next = common;

struct Node \*temp2 = head2;

while(temp2->next!=NULL){

temp2=temp2->next;

}

temp2->next = common;

struct Node \*intersection = findinter(head,head2);

printf("%d",intersection->data);\*/

display(head);

printf("\n");

int k =2;

//struct Node \*left=leftrotate(head,k);

struct Node \*right =rightrotate(head,k);

//display(left); //1 2 3 2 1

display(right);

}