Assignment 3

Write a C program to implement the following functions:

1. Reverse a linked list:

Function: Node* reverseList(Node* head)

Given a pointer to the head node of a linked list, the task is to reverse the linked list. We need to reverse the list by changing the links between nodes.

2. Remove duplicates from a linked list:

Function: void removeDuplicates(Node* head)

Input: linked list = 12− > 11− > 12− > 21− > 41− > 43− > 21

Output: 12− > 11− > 21− > 41− > 43

Explanation: Second occurrence of 12 and 21 is removed

3. Find the middle element of a linked list:

Function: Node* findMiddle(Node* head)

Given a singly linked list, find the middle of the linked list. For example, if the given linked list is 1− > 2− > 3− > 4− > 5 then the output should be 3. If there are even nodes, then there would be two middle nodes, we need to print the second middle element.

For example, if the given linked list is 1− > 2− > 3− > 4− > 5− > 6 then the output should be 4.

4. Check if a linked list is palindrome:

Function: bool isPalindrome(Node* head)

Input: Original Linked List: 1 2 2 1

Output: Linked list is a palindrome.

CODE

```
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
typedef struct node{
    int data;
    struct node* next;

}Node_t,*Node;
Node newNode(int data,Node next)
{
    Node node=(Node)malloc(sizeof(Node_t));
    node->data=data;
    node->next=next;
    return node;
}
void add_element(Node *pnode,int data)
{
    if(*pnode==NULL)
    {
        *pnode=newNode(data,*pnode);
        return;
    }
    Node node=*pnode;
    while(node->next!=NULL)
    {
        node=node->next;
    }
    node->next=newNode(data,node->next);

}
void display(Node pnode)
{
    while(pnode!=NULL)
    {
        printf("%d\t",pnode->data);
        pnode=pnode->next;
    }
    printf("\n");
}
void Reverse(Node *pnode)
{
    if((*pnode)->next==NULL)
    {
        return;
    }
    Node prev=NULL;
    Node curr=*pnode;
    Node NEXT=NULL;
    while(curr!=NULL)
```

```
    {

        NEXT=curr->next;
        curr->next=prev;
        prev=curr;
        curr=NEXT;
    }
    *pnode=prev;

}
void remove_duplicates(Node *pnode)
{
    Node node=*pnode;
    if(node==NULL || node->next==NULL)
    {
        printf("no duplicates\n");
        return;
    }

    while(node!=NULL)
    {
        Node curr=node->next;
        Node prev=node;
        while(curr!=NULL)
        {
            if(node->data==curr->data)
            {
                prev->next=curr->next;
                Node temp=curr;
                curr=curr->next;
                temp->next=NULL;
                free(temp);

            }
            else{
                prev=prev->next;
                curr=curr->next;
            }
        }
        node=node->next;
    }
}
Node find_middle(Node *pnode)
{
    Node node=*pnode;
    if(node==NULL || node->next==NULL)
    {
        return node;
    }
    if(node->next->next==NULL)
    {
```

```
        return node->next;
    }
    Node slow=node;
    Node fast=node->next;
    while(fast!=NULL)
    {
        fast=fast->next;
        slow=slow->next;
        if(fast!=NULL)
        {
            fast=fast->next;
        }


    }

    return slow;


}
bool ispallindrome(Node *pnode)
{
    if((*pnode)->next==NULL)
    {
        return true;
    }
    int i=0;
    Node node=*pnode;
    Node head=find_middle(pnode);
    Reverse(&head);
    while(head->next!=NULL)
    {
        if(node->data!=head->data)
        {
            return false;
        }
        node=node->next;
        head=head->next;
    }
    return true;

}
int main()
{
    Node head=NULL;
    int n;
    printf("enter number of nodes:");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        int ele;
```

```
        printf("enter element %d:",i+1);
        scanf("%d",&ele);
        add_element(&head,ele);
    }
    display(head);
    int num;
    Node x=NULL;
    bool y;
    printf("enter 1 for reverse\n");
    printf("enter 2 for remove_duplicates\n");
    printf("enter 3 for middle\n");
    printf("enter 4 for pallindrome check\n");
    scanf("%d",&num);
    switch(num)
    {
        case 1:Reverse(&head);
               display(head);
               break;
        case 2:remove_duplicates(&head);
               display(head);
               break;
        case 3:x=find_middle(&head);
               printf("middle element is =%d \n",x->data);
               break;
        case 4:y=ispallindrome(&head);
               if(y==true)
               {
                printf("IT IS PALLINDROME");

               }
               else
               {
                printf("NOT pallindrome");
               }
               break;
    }


}
```