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:

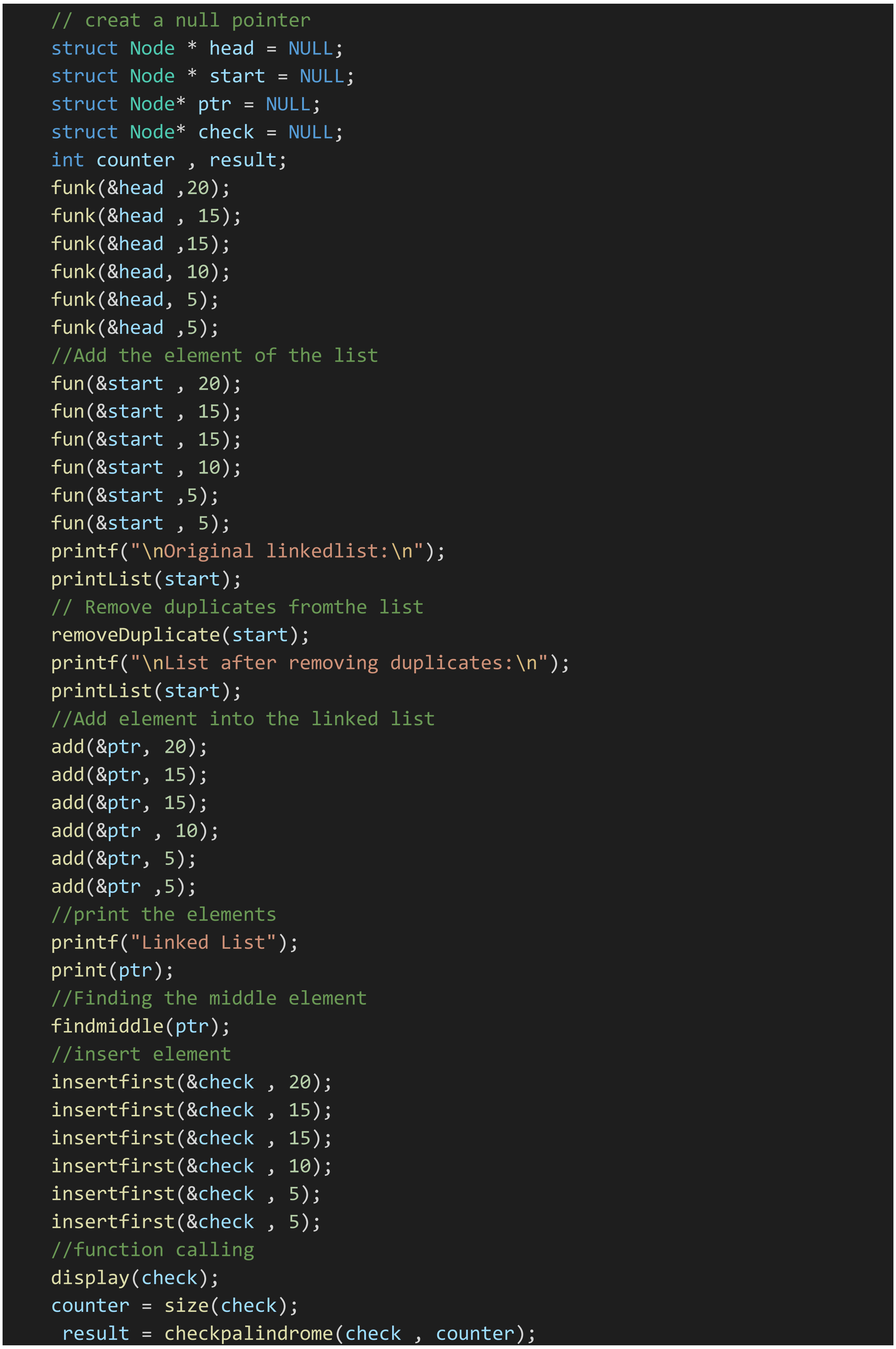
Function: bool isPalindrome(Node* head)

Input: Original Linked List: 1 2 2 1

Output: Linked list is a palindrome.

```c
# include<stdio.h>
# include<stdlib.h>
//Link list node
struct Node {
    int data;
    struct Node* link;
};
//function definition to reverse a linklist
static void reverse(struct Node** head_ref){
    struct Node* prev = NULL;
    struct Node* current = *head_ref;
    struct Node* link = NULL;
    while (current != NULL){
    //store next
     link= current->link;
    //Reverse current node pointer
    current ->link = prev;
    // move pointers one position ahead .
    prev = current;
    current = link;
    }
    *head_ref = prev;

}
void funk(struct Node**head_ref , int newdata){
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = newdata;
    new_node->link = (*head_ref);
        (*head_ref)=new_node;
}
        // Function to print linked list
        void printLists(struct Node*head){
            struct Node*temp = head;
            while(temp !=NULL){
                printf("%d", temp->data);
                temp = temp->link;
            }
        }
        //* Removing duplicate element from linklist
        //Function to add new node of linkedlist.
        void fun(struct Node**start_ref , int new_data){
            struct Node*second_node = (struct Node*)malloc(sizeof(struct
Node));
            second_node-> data = new_data;
            second_node->link = (*start_ref);
            (*start_ref)= second_node;
        }
        //function to remove duplicate element from linked list
        void removeDuplicate(struct Node*start){
            struct Node* current = start;
            struct Node* temp;
            if(current == NULL)return;
          //Traverse the linked list
          while(current->link !=NULL){
            // If duplicate found, delete
            if(current->data == current->link->data){
                temp = current->link->link;
                free(current->link);
                current->link  = temp;
            }
            else{
                current = current->link;
               }
            }
        }
        //Function to print the link list
        void printList(struct Node*node){
            while(node!=NULL){
                printf(" %d ",node->data);
                node= node->link;
            }
            printf("\n");
        }
        //function for finding middle element of the linklist
        void findmiddle(struct Node*str){
            struct Node*slow_ptr = str;
            struct Node*fast_ptr = str;

            if(str!=NULL){
                while(fast_ptr !=NULL && fast_ptr->link !=NULL){
                    fast_ptr = fast_ptr->link->link;
                    slow_ptr = slow_ptr->link;
                }
                printf("The middle element is %d\n",slow_ptr->data);
            }
        }
        //Function to insert a node at the end of a linked list
        void add(struct Node** ptr_ref, int new){
            struct Node*new_node = (struct Node*)malloc(sizeof(struct Node));
            struct Node*last = *ptr_ref;
            new_node ->data = new;
            new_node ->link = NULL;

            if(*ptr_ref == NULL){
                *ptr_ref = new_node;
                return;
            }
            while(last->link!=NULL){
            last= last->link;
            last->link = new_node;
            return;
            }
            //Function to print the elements of a linked list
            void print(struct Node* middlenode){
                while(middlenode != NULL){
                    printf("%d",middlenode->data);
                    middlenode = middlenode->link;
                }
                printf("NULL\n");
            }
        }
```

```
        void insertfirst(struct Node**check, int data){
            struct Node * addresh = (struct Node*)malloc(sizeof(struct Node));
            //assigning data value
            addresh->data = data;
            addresh->link = *check;
            //re-assign check to this new node
            *check = addresh;
        }
        void display (struct Node *node){
            printf("Linked list :\n");
            //as linked is end when node is null
            while(node != NULL){
                printf("%d",node->data);
                node = node->link;
            }
            printf("\n");
        }
        int size (struct Node*node){
            int counter =0;
            //as linkedlist will end when node is Null
            while(node != NULL){
                node =node->link;
                counter++;
            }
            return counter;
        }
        int checkpalindrome(struct Node *check , int counter){
            int i=0, j;
            struct Node*front,*rear;
            while(i!=counter/2){
                front =rear = check;
                for(j =0; j<i; j++){
                    front = front->link;
                }
                for(j =0; j<counter -(i+1); j++){
                    rear = rear->link;
                }
                if(front->data !=rear->data){
                    return 0;
                }
                else{
                    i++;
                }
            }
            return 1;
        }

int main(){
```



```c
    // creat a null pointer
    struct Node * head = NULL;
    struct Node * start = NULL;
    struct Node* ptr = NULL;
    struct Node* check = NULL;
    int counter , result;
    funk(&head ,20);
    funk(&head , 15);
    funk(&head ,15);
    funk(&head, 10);
    funk(&head, 5);
    funk(&head ,5);
    //Add the element of the list
    fun(&start , 20);
    fun(&start , 15);
    fun(&start , 15);
    fun(&start , 10);
    fun(&start ,5);
    fun(&start , 5);
    printf("\nOriginal linkedlist:\n");
    printList(start);
    // Remove duplicates fromthe list
    removeDuplicate(start);
    printf("\nList after removing duplicates:\n");
    printList(start);
    //Add element into the linked list
    add(&ptr, 20);
    add(&ptr, 15);
    add(&ptr, 15);
    add(&ptr , 10);
    add(&ptr, 5);
    add(&ptr ,5);
    //print the elements
    printf("Linked List");
    print(ptr);
    //Finding the middle element
    findmiddle(ptr);
    //insert element
    insertfirst(&check , 20);
    insertfirst(&check , 15);
    insertfirst(&check , 15);
    insertfirst(&check , 10);
    insertfirst(&check , 5);
    insertfirst(&check , 5);
    //function calling
    display(check);
    counter = size(check);
     result = checkpalindrome(check , counter);
     if(result == 1){
        printf("The linked list is a palindrome.\n");
     }
     else{
        printf("The linkedlist is not a palindrome. \n");
     }
return 0;
}
```