Symbiosis Institute of Technology

Faculty of Engineering

CSE- Academic Year 2024-25

Data Structures – Lab Batch 2023-27

Lab Assignment No:- 1

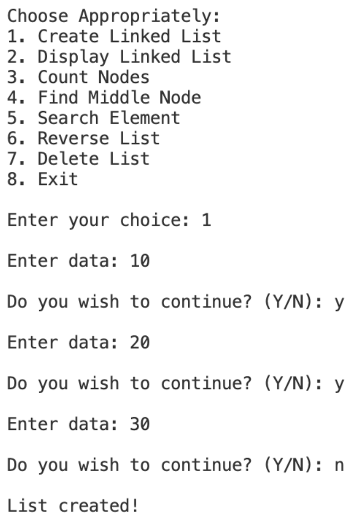
Name of Student	Faheemuddin Sayyed
PRN No.	23070122196
Batch	23-27
Class	CSE C-1
Academic Year & Semester	SY 24-25
Date of Performance	22/08/24
Title of Assignment:	• Creation of Singly Linked list for integers, Characters and Strings • Count Number of Nodes • Find the middle of the linked list. If the number of nodes are even, then there would be two middle nodes, so return the second middle node. • Searching element in Singly Linked list

Source Code/Algorithm/Flow Chart:

```
#include <stdio.h>
#include <stdlib.h>

struct Node{
    int data;
    struct Node *next;
}*first = NULL, *last = NULL;

typedef struct Node node;

void create(){
    node *p;
    int data;
    char choice;

    do{
        p = (node *)malloc(sizeof(node));
        printf("\nEnter data: ");
        scanf("%d", &data);
        getchar();

        if(!first){
            p -> data = data;
            p -> next = NULL;
            first = p;
            last = p;
        } else {
            p -> data = data;
            p -> next = NULL;
            last -> next = p;
            last = p;
        }

        printf("\nDo you wish to continue? (Y/N): ");
        scanf("%c", &choice);
    } while(choice == 'y' || choice == 'Y');
    printf("\nList created!\n");
}

void display(){
    node *p = first;

    printf("\n");
    while(p){
        printf("%d ", p -> data);
        p = p -> next;
    }
    printf("\n");
}

int count(){
    node *p = first;
    int count = 0;

    while(p){
        count++;
        p = p -> next;
    }
    return count;
}

int middleNode(){
    node *p = first;
    int len = count();
    for(int i = 0; i < (len / 2); i++) p = p -> next;
    return p -> data;
}

void search(int num){
    node *p = first;

    for(int i = 0; i < count(); i++){
        if(p -> data == num){
            printf("\nElement found at %d position.\n", i);
            return;
        }
        p = p -> next;
    }

    printf("\nElement not found!\n");
}

void reverse(node *p){
    if(!p -> next){
        first = p;
        return;
    }

    reverse(p -> next);
    node *q = p -> next;
    q -> next = p;
    p -> next = NULL;
}

void delete(){
    node *p = first;
    node *q;

    while(p){
        q = p -> next;
        free(p);
        p = q;
    }
    first = NULL;
}

int main(){
    int choice, num, exit = 1;

    while(exit){
        printf("\nChoose Appropriately:\
        \n1. Create Linked List\
        \n2. Display Linked List\
        \n3. Count Nodes\
        \n4. Find Middle Node\
        \n5. Search Element\
        \n6. Reverse List\
        \n7. Delete List\
        \n8. Exit\n");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        getchar();

        switch(choice){
            case 1:
                if(first){
                    printf("\nDelete previous list first!");
                }
                else{
                    create();
                }
                break;
            case 2:
                display();
                break;
            case 3:
                printf("\nNumber of nodes: %d\n", count());
                break;
            case 4:
                printf("\nMiddle node: %d\n", middleNode());
                break;
            case 5:
                printf("\nEnter element to search: ");
                scanf("%d", &num);
                getchar();
                search(num);
                break;
            case 6:
                reverse(first);
                printf("\nList reversed\n");
                break;
            case 7:
                delete();
                printf("\nList deleted!\n");
                break;
            case 8:
                exit = 0;
                break;
            default:
                printf("\nInvalid choice!\n");
        }
    }
    return 0;
}
```

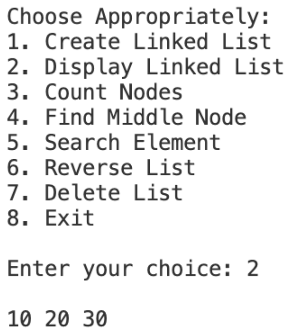
Output Screenshots

1) Linked List Creation:



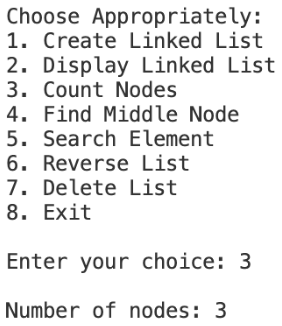
```
Choose Appropriately:
1. Create Linked List
2. Display Linked List
3. Count Nodes
4. Find Middle Node
5. Search Element
6. Reverse List
7. Delete List
8. Exit

Enter your choice: 1

Enter data: 10

Do you wish to continue? (Y/N): y

Enter data: 20

Do you wish to continue? (Y/N): y

Enter data: 30

Do you wish to continue? (Y/N): n

List created!
```

2) Display Linked List:



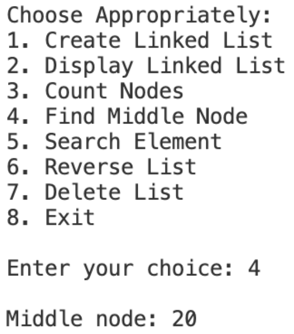
```
Choose Appropriately:
1. Create Linked List
2. Display Linked List
3. Count Nodes
4. Find Middle Node
5. Search Element
6. Reverse List
7. Delete List
8. Exit

Enter your choice: 2

10 20 30
```

3) Display node count:



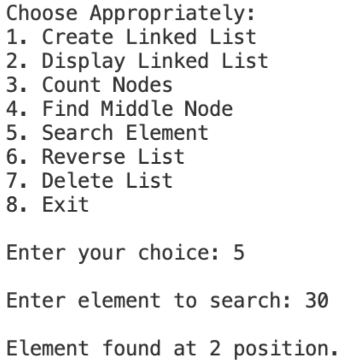
```
Choose Appropriately:
1. Create Linked List
2. Display Linked List
3. Count Nodes
4. Find Middle Node
5. Search Element
6. Reverse List
7. Delete List
8. Exit

Enter your choice: 3

Number of nodes: 3
```

4) Finding middle node:



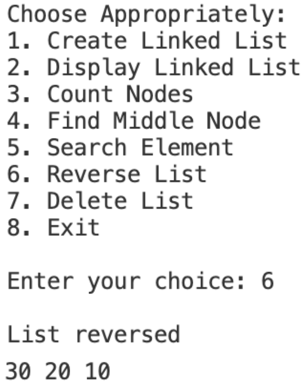
```
Choose Appropriately:
1. Create Linked List
2. Display Linked List
3. Count Nodes
4. Find Middle Node
5. Search Element
6. Reverse List
7. Delete List
8. Exit

Enter your choice: 4

Middle node: 20
```

5) Searching for an element:



```
Choose Appropriately:
1. Create Linked List
2. Display Linked List
3. Count Nodes
4. Find Middle Node
5. Search Element
6. Reverse List
7. Delete List
8. Exit

Enter your choice: 5

Enter element to search: 30

Element found at 2 position.
```

6) Reversing Linked List:



```
Choose Appropriately:
1. Create Linked List
2. Display Linked List
3. Count Nodes
4. Find Middle Node
5. Search Element
6. Reverse List
7. Delete List
8. Exit

Enter your choice: 6

List reversed
30 20 10
```

Practice questions	N/A
Conclusion	Thus we have explored all the different singly linked list operations and implemented them using a menu driven program.