

# Rajalakshmi Engineering College

Name: Aishwarya R  
Email: 241501011@rajalakshmi.edu.in  
Roll no: 241501011  
Phone: null  
Branch: REC  
Department: I AI & ML FA  
Batch: 2028  
Degree: B.E - AI & ML

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 1\_COD\_Question 7

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Dev is tasked with creating a program that efficiently finds the middle element of a linked list. The program should take user input to populate the linked list by inserting each element into the front of the list and then determining the middle element.

Assist Dev, as he needs to ensure that the middle element is accurately identified from the constructed singly linked list:

If it's an odd-length linked list, return the middle element. If it's an even-length linked list, return the second middle element of the two elements.

##### ***Input Format***

The first line of input consists of an integer n, representing the number of elements in the linked list.

The second line consists of n space-separated integers, representing the elements of the list.

### **Output Format**

The first line of output displays the linked list after inserting elements at the front.

The second line displays "Middle Element: " followed by the middle element of the linked list.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

10 20 30 40 50

Output: 50 40 30 20 10

Middle Element: 30

### **Answer**

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};

// You are using GCC
Node* createnode(int data){
    Node* newnode=(Node*)malloc(sizeof(data));
    if(newnode!=NULL){
        newnode->data=data;
        newnode->next=NULL;
        return newnode;
    }
    return NULL;
}
```

```
Node* push(Node* head,int data)
{
    Node* newnode=createNode(data);
    if(newnode==NULL)
    {
        printf("Memory allocation failed\n");
        exit(1);
    }

    newnode->next=head;
    return newnode;
}
```

```
int printMiddle(Node* head)
{
    if(head!=NULL)
    {
        Node*slow=head;
        Node*fast=head;

        while(fast!=NULL &&fast->next!=NULL)
        {
            slow=slow->next;
            fast=fast->next->next;
        }
        return slow->data;
    }
}
```

```
int main() {
    struct Node* head = NULL;
    int n;

    scanf("%d", &n);
    int value;

    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        head = push(head, value);
    }
}
```

```
struct Node* current = head;
while (current != NULL) {
    printf("%d ", current->data);
    current = current->next;
}
printf("\n");
```

```
int middle_element = printMiddle(head);
printf("Middle Element: %d\n", middle_element);
```

```
current = head;
while (current != NULL) {
    struct Node* temp = current;
    current = current->next;
    free(temp);
}

return 0;
}
```

**Status :** Correct

**Marks :** 10/10