# Rajalakshmi Engineering College

Name: Karthikeyan M
Email: 240801150@rajalakshmi.edu.in
Roll no: 2116240801150
Phone: 8056008890
Branch: REC
Department: I ECE FB
Batch: 2028
Degree: B.E - ECE

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;
};

struct Node* push(struct Node* head_ref, int new_data) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    if (new_node == NULL) {
        perror("Failed to allocate memory for new node");
        exit(EXIT_FAILURE);
    }
    new_node->data = new_data;
    new_node->next = head_ref;
    return new_node;
}
```

```
int printMiddle(struct Node* head) {
    if (head == NULL) {
        return -1;
    }

    struct Node* slow_ptr = head;
    struct Node* fast_ptr = head;
    int count = 0;

    while (fast_ptr != NULL && fast_ptr->next != NULL) {
        fast_ptr = fast_ptr->next->next;
        slow_ptr = slow_ptr->next;
        count += 2;
    }


    if (fast_ptr != NULL) {
        count++;
    }


    return slow_ptr->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**