Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Your task is to create a program to manage a playlist of items. Each item is represented as a character, and you need to implement the following operations on the playlist.

Here are the main functionalities of the program:

Insert Item: The program should allow users to add items to the front and end of the playlist. Items are represented as characters.Display Playlist: The program should display the playlist containing the items that were added.

To implement this program, a doubly linked list data structure should be used, where each node contains an item character.

Input Format

The input consists of a sequence of space-separated characters, representing the items to be inserted into the doubly linked list.

The input is terminated by entering - (hyphen).

Output Format

The first line of output prints "Forward Playlist: " followed by the linked list after inserting the items at the end.

The second line prints "Backward Playlist: " followed by the linked list after inserting the items at the front.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a b c -

Output: Forward Playlist: a b c
Backward Playlist: c b a

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Moniksha, a chess coach organizing a tournament, needs a program to manage participant IDs efficiently. The program maintains a doubly linked list of IDs and offers two functions: Append to add IDs as students register, and Print Maximum ID to identify the highest ID for administrative tasks.

This tool streamlines tournament organization, allowing Moniksha to focus on coaching her students effectively.

Input Format

The first line consists of an integer n, representing the number of participant IDs to be added.

The second line consists of n space-separated integers representing the participant IDs.

Output Format

The output displays a single integer, representing the maximum participant ID.

If the list is empty, the output prints "Empty list!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
163 137 155

Output: 163

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>


struct Node {
   int id;
   struct Node* next;
   struct Node* prev;
};


struct Node* createNode(int id) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->id = id;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}


void append(struct Node** head, int id) {
   struct Node* newNode = createNode(id);
   if (*head == NULL) {
      *head = newNode;
```

```
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}


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

    int maxId = head->id;
    struct Node* current = head->next;
    while (current != NULL) {
        if (current->id > maxId) {
            maxId = current->id;
        }
        current = current->next;
    }
    return maxId;
}


int main() {
    int n;
    scanf("%d", &n);

    if (n == 0) {
        printf("Empty list!\n");
        return 0;
    }

    struct Node* head = NULL;


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

    int maxId = findMax(head);
    if (maxId == -1) {
        printf("Empty list!\n");
    } else {
        printf("%d\n", maxId);
    }


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

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Bob is tasked with developing a company's employee record management system. The system needs to maintain a list of employee records using a doubly linked list. Each employee is represented by a unique integer ID.

Help Bob to complete a program that adds employee records at the front, traverses the list, and prints the same for each addition of employees to the list.

Input Format

The first line of input consists of an integer N, representing the number of employees.

The second line consists of N space-separated integers, representing the employee IDs.

Output Format

For each employee ID, the program prints "Node Inserted" followed by the current state of the doubly linked list in the next line, with the data values of each node separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 4
101 102 103 104
Output: Node Inserted
101
Node Inserted
102 101
Node Inserted
103 102 101
Node Inserted
104 103 102 101
```

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is developing a student registration system for a college. To efficiently store and manage the student IDs, he decides to implement a doubly linked list where each node represents a student's ID.

In this system, each student's ID is stored sequentially, and the system needs to display all registered student IDs in the order they were entered.

Implement a program that creates a doubly linked list, inserts student IDs, and displays them in the same order.

Input Format

The first line contains an integer N the number of student IDs.

The second line contains N space-separated integers representing the student IDs.

Output Format

The output should display the single line containing N space-separated integers representing the student IDs stored in the doubly linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 10 20 30 40 50

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>


struct Node {
   int data;
   struct Node* next;
   struct Node* prev;
};


struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}


void insertEnd(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}


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

int main() {
    int N;
    scanf("%d", &N);

    struct Node* head = NULL;

    for (int i = 0; i < N; i++) {
        int id;
        scanf("%d", &id);
        insertEnd(&head, id);
    }

    display(head);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Ashiq is developing a ticketing system for a small amusement park. The park issues tickets to visitors in the order they arrive. However, due to a system change, the oldest ticket (first inserted) must be revoked instead of the last one.

To manage this, Ashiq decided to use a doubly linked list-based stack, where:

Pushing adds a new ticket to the top of the stack.Removing the first inserted ticket (removing from the bottom of the stack).Printing the remaining tickets from bottom to top.

Input Format

The first line consists of an integer n, representing the number of tickets issued.

The second line consists of n space-separated integers, each representing a ticket number in the order they were issued.

Output Format

The output prints space-separated integers, representing the remaining ticket numbers in the order from bottom to top.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
24 96 41 85 97 91 13
Output: 96 41 85 97 91 13

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
   int ticket;
   struct Node* prev;
   struct Node* next;
} Node;
void push(Node** head_ref, Node** tail_ref, int ticket) {
   Node* new_node = (Node*)malloc(sizeof(Node));
   new_node->ticket = ticket;
   new_node->next = NULL;
   new_node->prev = NULL;

   if (*head_ref == NULL) {
      *head_ref = new_node;
      *tail_ref = new_node;
   } else {
      (*tail_ref)->next = new_node;
      new_node->prev = *tail_ref;
      *tail_ref = new_node;
   }
}
```

```
void remove_bottom(Node** head_ref, Node** tail_ref) {
  if (*head_ref == NULL) return;

  Node* temp = *head_ref;
  *head_ref = temp->next;

  if (*head_ref != NULL)
    (*head_ref)->prev = NULL;
  else
    *tail_ref = NULL;

  free(temp);
}
void print_list(Node* head) {
  Node* current = head;
  while (current != NULL) {
    printf("%d ", current->ticket);
    current = current->next;
  }
}

int main() {
  int n, i, ticket;
  Node *head = NULL, *tail = NULL;

  scanf("%d", &n);
  for (i = 0; i < n; i++) {
    scanf("%d", &ticket);
    push(&head, &tail, ticket);
  }
  remove_bottom(&head, &tail);
  print_list(head);
  printf("\n");
  while (head != NULL) {
    remove_bottom(&head, &tail);
  }

  return 0;
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Aarav is working on a program to analyze his test scores, which are stored in a doubly linked list. He needs a solution to input scores into the list and determine the highest score.

Help him by providing code that lets users enter test scores into the doubly linked list and find the maximum score efficiently.

Input Format

The first line consists of an integer N, representing the number of elements to be initially inserted into the doubly linked list.

The second line consists of N space-separated integers, denoting the score to be inserted.

Output Format

The output prints an integer, representing the highest score present in the list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
89 71 2 70
Output: 89

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
   int score;
   struct Node* prev;
   struct Node* next;
} Node;
void insert_end(Node** head_ref, Node** tail_ref, int score) {
   Node* new_node = (Node*)malloc(sizeof(Node));
```

```
    new_node->score = score;
    new_node->next = NULL;
    new_node->prev = NULL;

    if (*head_ref == NULL) {
        *head_ref = new_node;
        *tail_ref = new_node;
    } else {
        (*tail_ref)->next = new_node;
        new_node->prev = *tail_ref;
        *tail_ref = new_node;
    }
}
int find_max(Node* head) {
    if (head == NULL) return -1;

    int max = head->score;
    Node* current = head->next;

    while (current != NULL) {
        if (current->score > max)
            max = current->score;
        current = current->next;
    }
    return max;
}
int main() {
    int N, i, score;
    Node* head = NULL;
    Node* tail = NULL;

    scanf("%d", &N);

    for (i = 0; i < N; i++) {
        scanf("%d", &score);
        insert_end(&head, &tail, score);
    }

    int max_score = find_max(head);
    printf("%d\n", max_score);

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

    return 0;
}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Sam is learning about two-way linked lists. He came across a problem where he had to populate a two-way linked list and print the original as well as the reverse order of the list. Assist him with a suitable program.

Input Format

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

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

Output Format

The first line displays the message: "List in original order:"

The second line displays the elements of the doubly linked list in the original order.

The third line displays the message: "List in reverse order:"

The fourth line displays the elements of the doubly linked list in reverse order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
Output: List in original order:
1 2 3 4 5
List in reverse order:
5 4 3 2 1

Answer

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node
{
   int data;
   struct Node* prev;
   struct Node* next;
} Node;
int main()
{
   int n;
   scanf("%d", &n);
   Node *head = NULL, *tail = NULL;
   for (int i = 0; i < n; i++)
{
      int val;
      scanf("%d", &val);
      Node* newNode = (Node*)malloc(sizeof(Node));
      newNode->data = val;
      newNode->prev = tail;
      newNode->next = NULL;
      if (tail) tail->next = newNode;
      else head = newNode;
      tail = newNode;
}
   printf("List in original order:\n");
   for (Node* cur = head; cur != NULL; cur = cur->next)
      printf("%d ", cur->data);
   printf("\n");
   printf("List in reverse order:\n");
   for (Node* cur = tail; cur != NULL; cur = cur->prev)
      printf("%d ", cur->data);
   printf("\n");
   Node* cur = head;
```

```
    while (cur)
{
        Node* temp = cur;
        cur = cur->next;
        free(temp);
}
    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Tom is a software developer working on a project where he has to check if a doubly linked list is a palindrome. He needs to write a program to solve this problem. Write a program to help Tom check if a given doubly linked list is a palindrome or not.

Input Format

The first line 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 linked list elements.

Output Format

The first line displays the space-separated integers, representing the doubly linked list.

The second line displays one of the following:

1. If the doubly linked list is a palindrome, print "The doubly linked list is a palindrome".
2. If the doubly linked list is not a palindrome, print "The doubly linked list is not a palindrome".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 2 1

Output: 1 2 3 2 1
The doubly linked list is a palindrome

Answer

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node
{
   int data;
   struct Node *prev, *next;
} Node;
int main()
{
   int n; scanf("%d", &n);
   Node *head = NULL, *tail = NULL;
   for (int i = 0; i < n; i++)
{
      int val; scanf("%d", &val);
      Node *newNode = (Node*)malloc(sizeof(Node));
      newNode->data = val;
      newNode->next = NULL;
      newNode->prev = tail;
      if (tail) tail->next = newNode;
      else head = newNode;
      tail = newNode;
}
   Node *cur = head;
   while (cur)
{
      printf("%d ", cur->data);
      cur = cur->next;
}
   printf("\n");
   Node *front = head, *back = tail;
   int palindrome = 1;
   while (front != back && front->prev != back)
{
      if (front->data != back->data)
{
         palindrome = 0;
         break;
}
      front = front->next;
      back = back->prev;
}
   if (palindrome)
      printf("The doubly linked list is a palindrome\n");
   else
      printf("The doubly linked list is not a palindrome\n");
   cur = head;
   while (cur)
{
      Node *temp = cur;
      cur = cur->next;
      free(temp);
}
   return 0;
}
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Rohan is a software developer who is working on an application that processes data stored in a Doubly Linked List. He needs to implement a feature that finds and prints the middle element(s) of the list. If the list contains an odd number of elements, the middle element should be printed. If the list contains an even number of elements, the two middle elements should be printed.

Help Rohan by writing a program that reads a list of numbers, prints the list, and then prints the middle element(s) based on the number of elements in the list.

Input Format

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

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

Output Format

The first line prints the elements of the list separated by space. (There is an extra space at the end of this line.)

The second line prints the middle element(s) based on the number of elements.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
20 52 40 16 18

Output: 20 52 40 16 18
40

Answer

```
#include <stdio.h>
int main()
{
   int n;
   scanf("%d", &n);
   int arr[10];
   for (int i = 0; i < n; i++)
{
    scanf("%d", &arr[i]);
}
   for (int i = 0; i < n; i++)
{
    printf("%d ", arr[i]);
}
   printf("\n");
   if (n % 2 == 1)
{
    printf("%d\n", arr[n / 2]);
} else
{
    printf("%d %d\n", arr[(n / 2) - 1], arr[n / 2]);
}
   return 0;
}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Pranav wants to clockwise rotate a doubly linked list by a specified number of positions. He needs your help to implement a program to achieve this. Given a doubly linked list and an integer representing the number of positions to rotate, write a program to rotate the list clockwise.

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 linked list elements.

The third line consists of an integer k, representing the number of places to rotate the list.

Output Format

The output displays the elements of the doubly linked list after rotating it by k positions.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
1

Output: 5 1 2 3 4

Answer

```
#include <stdio.h>
int main()
{
   int n, k;
   scanf("%d", &n);
   int arr[30];
   for(int i = 0; i < n; i++)
{
      scanf("%d", &arr[i]);
}
   scanf("%d", &k);
   for(int i = n - k; i < n; i++)
{
      printf("%d ", arr[i]);
}
   for(int i = 0; i < n - k; i++)
{
      printf("%d ", arr[i]);
}
   printf("\n");
   return 0;
}
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Riya is developing a contact management system where recently added contacts should appear first. She decides to use a doubly linked list to store contact IDs in the order they are added. Initially, new contacts are inserted at the front of the list. However, sometimes she needs to insert a new contact at a specific position in the list based on priority.

Help Riya implement this system by performing the following operations:

Insert contact IDs at the front of the list as they are added.Insert a new contact at a given position in the list.

Input Format

The first line of input consists of an integer N, representing the initial size of the linked list.

The second line consists of N space-separated integers, representing the values of the linked list to be inserted at the front.

The third line consists of an integer position, representing the position at which the new value should be inserted (position starts from 1).

The fourth line consists of integer data, representing the new value to be inserted.

Output Format

The first line of output prints the original list after inserting initial elements to the front.

The second line prints the updated linked list after inserting the element at the specified position.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
10 20 30 40
3
25
Output: 40 30 20 10
40 30 25 20 10

Answer

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node
{
   int data;
   struct Node* prev;
   struct Node* next;
} Node;
Node* insertFront(Node* head, int data)
{
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = head;
   if (head) head->prev = newNode;
   return newNode;
}
Node* insertAtPosition(Node* head, int pos, int data)
{
   if (pos == 1) return insertFront(head, data);
   Node* temp = head;
   for (int i = 1; i < pos - 1 && temp != NULL; i++)
{
      temp = temp->next;
}
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->next = temp->next;
   newNode->prev = temp;
   if (temp->next) temp->next->prev = newNode;
   temp->next = newNode;
   return head;
}
void printList(Node* head)
{
   Node* temp = head;
   while (temp)
{
      printf("%d ", temp->data);
      temp = temp->next;
}
```

```
    printf("\n");
}
int main()
{
    int N, pos, data;
    scanf("%d", &N);
    Node* head = NULL;
    for (int i = 0; i < N; i++)
{
        int val;
        scanf("%d", &val);
        head = insertFront(head, val);
}
    scanf("%d %d", &pos, &data);
    printList(head);
    head = insertAtPosition(head, pos, data);
    printList(head);
    return 0;
}
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Bala is a student learning about the doubly linked list and its functionalities. He came across a problem where he wanted to create a doubly linked list by appending elements to the front of the list.

After populating the list, he wanted to delete the node at the given position from the beginning. Write a suitable code to help Bala.

Input Format

The first line contains an integer N, the number of elements in the doubly linked list.

The second line contains N integers separated by a space, the data values of the nodes in the doubly linked list.

The third line contains an integer X, the position of the node to be deleted from the doubly linked list.

Output Format

The first line of output displays the original elements of the doubly linked list, separated by a space.

The second line prints the updated list after deleting the node at the given position X from the beginning.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
2
Output: 50 40 30 20 10
50 30 20 10

Answer

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node
{
   int data;
   struct Node* prev;
   struct Node* next;
} Node;
Node* insertFront(Node* head, int data)
{
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = head;
   if (head) head->prev = newNode;
   return newNode;
}
Node* deleteAtPosition(Node* head, int pos)
{
   Node* temp = head;
   if (pos == 1)
{
        head = temp->next;
        if (head) head->prev = NULL;
        free(temp);
        return head;
}
    for (int i = 1; i < pos; i++)
{
        temp = temp->next;
}
    temp->prev->next = temp->next;
    if (temp->next) temp->next->prev = temp->prev;
    free(temp);
    return head;
}
void printList(Node* head)
{
    while (head)
{
        printf("%d ", head->data);
        head = head->next;
}
    printf("\n");
}
int main()
{
    int N, X, val;
    scanf("%d", &N);
    Node* head = NULL;
    for (int i = 0; i < N; i++)
{
        scanf("%d", &val);
        head = insertFront(head, val);
}
    scanf("%d", &X);
    printList(head);
    head = deleteAtPosition(head, X);
    printList(head);
    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: nila ela
Email: 241901074@rajalakshmi.edu.in
Roll no: 241901074
Phone: 9025155667
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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

struct Node* push(struct Node* head, int value) {
   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
   new_node->data = value;
   new_node->next = head;
   return new_node;
}


int printMiddle(struct Node* head) {
   struct Node* slow = head;
   struct 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***