Rajalakshmi Engineering College

Name: Akshayabalaji AKSHAYA
Email: 241801012@rajalakshmi.edu.in
Roll no: 241801012
Phone: 9080154501
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

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 : 2.5

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

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

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

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Function to append a node at the end of the list
void append(Node** head, Node** tail, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
    } else {
```

```
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}

// Function to find the maximum participant ID
int findMax(Node* head) {
    if (head == NULL) {
        printf("Empty list!\n");
        return -1;
    }

    int maxID = head->data;
    Node* current = head;

    while (current) {
        if (current->data > maxID) {
            maxID = current->data;
        }
        current = current->next;
    }

    return maxID;
}

// Function to print the participant IDs
void printList(Node* head) {
    while (head) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}

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

    // Read input
    scanf("%d", &n);
```

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

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

    // Print the participant IDs
    printList(head);

    // Print the maximum participant ID
    printf("%d\n", findMax(head));

    return 0;
}
// You are using GCC
```

Status : Partially correct **Marks : 2.5/10**

Rajalakshmi Engineering College

Name: Akshayabalaji AKSHAYA
Email: 241801012@rajalakshmi.edu.in
Roll no: 241801012
Phone: 9080154501
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

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

```
#include <iostream>
using namespace std;

struct node {
   int info;
   struct node* prev, * next;
};

struct node* start = NULL;


void traverse() {
   struct node*temp=start;
   while(temp!=NULL){
      printf(%d ",temp->info);
      temp=temp->next;
   }
   printf("\n");
}
```

```
void insertAtFront(int data) {
    struct node*nnode=(struct node*)malloc(sizeof(node));
    nnode->info=data;
    nnode->prev=NULL;
    nnode->next=start;

    if(start!=NULL){
        start->prev=nnode;
    }

    start=nnode;
    printf("Node Inserted\n");
}


// Function to create a new node

int main() {
    int n, data;
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cin >> data;
        insertAtFront(data);
        traverse();
    }
    return 0;
}
```

Status : Wrong **Marks : 0/10**

Rajalakshmi Engineering College

Name: Akshayabalaji AKSHAYA
Email: 241801012@rajalakshmi.edu.in
Roll no: 241801012
Phone: 9080154501
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

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 : 0

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

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

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

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Function to append a node at the end of the list
void append(Node** head, Node** tail, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
```

```
    } else {
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}

// Function to print the inventory list
void printList(Node* head) {
    Node* temp = head;
    int position = 1;
    printf("Data entered in the list:\n");
    while (temp) {
        printf(" node %d : %d\n", position++, temp->data);
        temp = temp->next;
    }
}

// Function to delete a node at the given position
void deleteAtPosition(Node** head, Node** tail, int position, int n) {
    if (position < 1 || position > n) {
        printf("Invalid position. Try again.\n");
        return;
    }

    Node* temp = *head;

    // If deleting the first node
    if (position == 1) {
        *head = temp->next;
        if (*head) (*head)->prev = NULL;
        free(temp);
    } else {
        for (int i = 1; i < position && temp; i++) {
            temp = temp->next;
        }

        if (temp == NULL) return;

        if (temp->prev) temp->prev->next = temp->next;
        if (temp->next) temp->next->prev = temp->prev;
        else *tail = temp->prev; // Update tail if last node is deleted
```

```
        free(temp);
    }

    // Print updated list
    Node* current = *head;
    printf("\nAfter deletion the new list:\n");
    int pos = 1;
    while (current) {
        printf(" node %d : %d\n", pos++, current->data);
        current = current->next;
    }
}

int main() {
    int n, p, value;
    Node* head = NULL;
    Node* tail = NULL;

    // Read input
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        append(&head, &tail, value);
    }

    // Print the original inventory list
    printList(head);

    // Read position to delete
    scanf("%d", &p);
    deleteAtPosition(&head, &tail, p, n);

    return 0;
}
// You are using GCC
```

Status : Wrong **Marks : 0/10**

Rajalakshmi Engineering College

Name: Akshayabalaji AKSHAYA
Email: 241801012@rajalakshmi.edu.in
Roll no: 241801012
Phone: 9080154501
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Ashwin is tasked with developing a simple application to manage a list of items in a shop inventory using a doubly linked list. Each item in the inventory has a unique identification number. The application should allow users to perform the following operations:

Create a List of Items: Initialize the inventory with a given number of items. Each item will be assigned a unique number provided by the user and insert the elements at end of the list.

Delete an Item: Remove an item from the inventory at a specific position.

Display the Inventory: Show the list of items before and after deletion.

If the position provided for deletion is invalid (e.g., out of range), it should

display an error message.

Input Format

The first line contains an integer n, representing the number of items to be initially entered into the inventory.

The second line contains n integers, each representing the unique identification number of an item separated by spaces.

The third line contains an integer p, representing the position of the item to be deleted from the inventory.

Output Format

The first line of output prints "Data entered in the list:" followed by the data values of each node in the doubly linked list before deletion.

If p is an invalid position, the output prints "Invalid position. Try again."

If p is a valid position, the output prints "After deletion the new list:" followed by the data values of each node in the doubly linked list after deletion.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1 2 3 4
5
Output: Data entered in the list:
node 1 : 1
node 2 : 2
node 3 : 3
node 4 : 4
Invalid position. Try again.

Answer

```
// Function to append a node at the end of the list
void append(Node** head, Node** tail, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
    } else {
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}

// Function to print the inventory list
void printList(Node* head) {
    Node* temp = head;
    int position = 1;
    printf("Data entered in the list:\n");
    while (temp) {
        printf(" node %d : %d\n", position++, temp->data);
        temp = temp->next;
    }
}

// Function to delete a node at the given position
void deleteAtPosition(Node** head, Node** tail, int position, int n) {
    if (position < 1 || position > n) {
        printf("Invalid position. Try again.\n");
        return;
    }

    Node* temp = *head;

    // If deleting the first node
    if (position == 1) {
        *head = temp->next;
        if (*head) (*head)->prev = NULL;
        free(temp);
    } else {
        for (int i = 1; i < position && temp; i++) {
            temp = temp->next;
```

```
        }

        if (temp == NULL) return;

        if (temp->prev) temp->prev->next = temp->next;
        if (temp->next) temp->next->prev = temp->prev;
        else *tail = temp->prev; // Update tail if last node is deleted

        free(temp);
    }

    // Print updated list
    Node* current = *head;
    printf("\nAfter deletion the new list:\n");
    int pos = 1;
    while (current) {
        printf(" node %d : %d\n", pos++, current->data);
        current = current->next;
    }
}
```

Status : Wrong **Marks : 0/10**