Rajalakshmi Engineering College

Name: VISHRUTHI K S
Email: 241801318@rajalakshmi.edu.in
Roll no: 241801318
Phone: 9791109021
Branch: REC
Department: I AI & DS FD
Batch: 2028
Degree: B.E - AI & DS

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 : 10

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

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

struct Node {
    char item;
    struct Node* next;
    struct Node* prev;
};

// You are using GCC
void insertAtEnd(struct Node** head, char item) {
   //type your code here
     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->item = item;
    newNode->next = NULL;

    if (*head == NULL) {
        newNode->prev = NULL;
        *head = newNode;
```

```
        return;
    }

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

    last->next = newNode;
    newNode->prev = last;

}
void displayForward(struct Node* head) {
    //type your code here
     struct Node* current = head;

    while (current != NULL) {
        printf("%c", current->item);
        if (current->next != NULL) {
            printf(" ");
        }
        current = current->next;
    }
    printf("\n");

}

void displayBackward(struct Node* tail) {
    //type your code here
      struct Node* current = tail;

    while (current != NULL) {
        printf("%c", current->item);
        if (current->prev != NULL) {
            printf(" ");
        }
        current = current->prev;
    }
    printf("\n");

}
```

```
void freePlaylist(struct Node* head) {
    //type your code here
     struct Node* current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }

}

int main() {
    struct Node* playlist = NULL;
    char item;

    while (1) {
        scanf(" %c", &item);
        if (item == '-') {
            break;
        }
        insertAtEnd(&playlist, item);
    }

    struct Node* tail = playlist;
    while (tail->next != NULL) {
        tail = tail->next;
    }

    printf("Forward Playlist: ");
    displayForward(playlist);

    printf("Backward Playlist: ");
    displayBackward(tail);

    freePlaylist(playlist);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: VISHRUTHI K S
Email: 241801318@rajalakshmi.edu.in
Roll no: 241801318
Phone: 9791109021
Branch: REC
Department: I AI & DS FD
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 : 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>
#include <limits.h>


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


struct Node* createNode(int id) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = id;
    newNode->prev = NULL;
    newNode->next = 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 INT_MIN;
    }
    int max = head->data;
    struct Node* current = head->next;
    while (current != NULL) {
        if (current->data > max) {
            max = current->data;
        }
        current = current->next;
    }
    return max;
}


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

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

    struct Node* head = NULL;
```

```
    if (n == 0) {
        printf("Empty list!");
    } else {
        for (int i = 0; i < n; i++) {
            scanf("%d", &id);
            append(&head, id);
        }
        int max_id = findMax(head);
        printf("%d", max_id);
        freeList(head);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: VISHRUTHI K S
Email: 241801318@rajalakshmi.edu.in
Roll no: 241801318
Phone: 9791109021
Branch: REC
Department: I AI & DS FD
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 : 10

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;

// You are using GCC
void traverse() {
   //type your code here
    node* temp = start;
    while (temp != NULL) {
        cout << temp->info;
        if (temp->next != NULL) {
            cout << " ";
        }
```

```
        temp = temp->next;
    }
    cout<<endl;

}

void insertAtFront(int data) {
  //type your code here
    node* newNode = new node();
    newNode->info = data;
    newNode->prev = NULL;
    newNode->next = start;

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

    cout << "Node Inserted" << endl;


}

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

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

Rajalakshmi Engineering College

Name: VISHRUTHI K S
Email: 241801318@rajalakshmi.edu.in
Roll no: 241801318
Phone: 9791109021
Branch: REC
Department: I AI & DS FD
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 : 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* prev;
    struct Node* next;
};


struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = 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 displayList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->next != NULL) {
            printf(" ");
        }
        temp = temp->next;
    }
}

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);
    }

    displayList(head);


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

```
        free(current);
        current = next;
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: VISHRUTHI K S
Email: 241801318@rajalakshmi.edu.in
Roll no: 241801318
Phone: 9791109021
Branch: REC
Department: I AI & DS FD
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 : 10

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

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

```
// Structure for a node in the doubly linked list
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

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

// Function to insert a node at the end of the list
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;
    }
}

// Function to delete a node at a specific position
void deleteAtPosition(struct Node** head, int position, int* success) {
    if (*head == NULL) {
        *success = 0;
        return;
    }

    struct Node* temp = *head;
```

```
    // Find the node at the given position
    for (int i = 1; temp != NULL && i < position; i++) {
        temp = temp->next;
    }

    // If position is out of range
    if (temp == NULL) {
        *success = 0;
        return;
    }

    // If node to be deleted is head node
    if (*head == temp) {
        *head = temp->next;
    }

    // Change next only if node to be deleted is NOT the last node
    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }

    // Change prev only if node to be deleted is NOT the first node
    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
    }

    free(temp);
    *success = 1;
}

// Function to display the list
void displayList(struct Node* head) {
    struct Node* temp = head;
    int count = 1;
    while (temp != NULL) {
        printf("node %d : %d\n", count++, temp->data);
        temp = temp->next;
    }
}

int main() {
    int n, position;
```

```
    scanf("%d", &n);

    struct Node* head = NULL;

    // Read and insert items
    for (int i = 0; i < n; i++) {
        int item;
        scanf("%d", &item);
        insertEnd(&head, item);
    }

    // Display initial list
    printf("Data entered in the list:\n");
    displayList(head);

    // Read position to delete
    scanf("%d", &position);

    // Delete node at position
    int success;
    deleteAtPosition(&head, position, &success);

    if (!success) {
        printf("Invalid position. Try again.\n");
    } else {
        // Display list after deletion
        printf("\nAfter deletion the new list:\n");
        displayList(head);
    }

    // Free the allocated memory
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
}
```

Status : Correct *Marks : 10/10*