

Rajalakshmi Engineering College

Name: Priyadharshan S
Email: 240801254@rajalakshmi.edu.in
Roll no: 240801254
Phone: 8124397424
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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;  
};
```

```
void insertAtEnd(struct Node** head, char item){  
    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* temp = *head;
```

```

    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=newnode;
    newnode->prev=temp;
}

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

void displayBackward(struct Node* tail){
    while(tail!=NULL){
        printf(" %c",tail->item);
        tail=tail->prev;
    }
    printf("\n");
}

void freePlaylist(struct Node* head){
    struct Node* temp;
    while(head!=NULL){
        temp=head;
        head=head->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: Priyadharshan S
Email: 240801254@rajalakshmi.edu.in
Roll no: 240801254
Phone: 8124397424
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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 data;  
    struct Node* next;  
    struct Node* prev;  
};
```

```
struct DoublyLinkedList {  
    struct Node* head;  
    struct Node* tail;  
};
```

```
struct DoublyLinkedList* initializeList() {  
    struct DoublyLinkedList* list = (struct DoublyLinkedList*)malloc(sizeof(struct  
DoublyLinkedList));  
    if (!list) {  
        perror("Memory allocation failed");  
        exit(EXIT_FAILURE);  
    }  
    list->head = NULL;  
    list->tail = NULL;  
    return list;
```

```
}
```

```
void append(struct DoublyLinkedList* list, int data) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    if (!newNode) {  
        perror("Memory allocation failed");  
        exit(EXIT_FAILURE);  
    }  
    newNode->data = data;  
    newNode->next = NULL;  
  
    if (!list->head) {  
        newNode->prev = NULL;  
        list->head = newNode;  
        list->tail = newNode;  
    } else {  
        newNode->prev = list->tail;  
        list->tail->next = newNode;  
        list->tail = newNode;  
    }  
}
```

```
void printMaximumID(struct DoublyLinkedList* list) {  
    if (!list->head) {  
        printf("Empty list!\n");  
        return;  
    }  
  
    int maximumID = -1;  
    struct Node* current = list->head;  
    while (current) {  
        if (current->data > maximumID) {  
            maximumID = current->data;  
        }  
        current = current->next;  
    }  
    printf("%d\n", maximumID);  
}
```

```
void freeList(struct DoublyLinkedList* list) {  
    struct Node* current = list->head;  
    struct Node* nextNode;
```

```

while (current) {
    nextNode = current->next;
    free(current);
    current = nextNode;
}
free(list);
}

int main() {
    int n;
    if (scanf("%d", &n) != 1) {
        fprintf(stderr, "Error reading the number of participants.\n");
        return 1;
    }

    struct DoublyLinkedList* participantList = initializeList();

    if (n > 0) {
        for (int i = 0; i < n; i++) {
            int id;
            if (scanf("%d", &id) != 1) {
                fprintf(stderr, "Error reading participant ID.\n");
                freeList(participantList);
                return 1;
            }
            append(participantList, id);
        }
        printMaximumID(participantList);
    } else {
        printMaximumID(participantList);
    }

    freeList(participantList);
    return 0;
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Priyadharshan S
Email: 240801254@rajalakshmi.edu.in
Roll no: 240801254
Phone: 8124397424
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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;
```

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

```
printf("\n");
```

```
}  
  
void insertAtFront(int data) {  
    Node* newnode=(Node*)malloc(sizeof(Node));  
    newnode->info=data;  
    newnode->next=NULL;  
    newnode->prev=NULL;  
    if(head==NULL){  
        head=newnode;  
    }  
    else{  
        newnode->next=head;  
        head->prev=newnode;  
        head=newnode;  
    }  
    printf("Node Inserted\n");  
}
```

```
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: Priyadharshan S

Email: 240801254@rajalakshmi.edu.in

Roll no: 240801254

Phone: 8124397424

Branch: REC

Department: I ECE AF

Batch: 2028

Degree: B.E - ECE

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 DoublyLinkedList {  
    struct Node* head;  
    struct Node* tail;  
};
```

```
struct DoublyLinkedList* initializeList() {  
    struct DoublyLinkedList* list = (struct DoublyLinkedList*)malloc(sizeof(struct  
DoublyLinkedList));  
    if (!list) {  
        perror("Memory allocation failed");  
        exit(EXIT_FAILURE);  
    }  
    list->head = NULL;  
    list->tail = NULL;
```

```

    return list;
}

void append(struct DoublyLinkedList* list, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->next = NULL;

    if (!list->head) {
        newNode->prev = NULL;
        list->head = newNode;
        list->tail = newNode;
    } else {
        newNode->prev = list->tail;
        list->tail->next = newNode;
        list->tail = newNode;
    }
}

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

void freeList(struct DoublyLinkedList* list) {
    struct Node* current = list->head;
    struct Node* nextNode;
    while (current) {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }
}

```

```

    }
    free(list);
}

int main() {
    int n;
    if (scanf("%d", &n) != 1) {
        fprintf(stderr, "Error reading the number of student IDs.\n");
        return 1;
    }

    struct DoublyLinkedList* studentList = initializeList();

    if (n > 0) {
        for (int i = 0; i < n; i++) {
            int id;
            if (scanf("%d", &id) != 1) {
                fprintf(stderr, "Error reading student ID.\n");
                freeList(studentList);
                return 1;
            }
            append(studentList, id);
        }
        printList(studentList);
    } else {
        printf("\n");
    }

    freeList(studentList);
    return 0;
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Priyadharshan S
Email: 240801254@rajalakshmi.edu.in
Roll no: 240801254
Phone: 8124397424
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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

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

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

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

```
struct DoublyLinkedList {  
    struct Node* head;  
    struct Node* tail;  
};
```

```
struct DoublyLinkedList* initializeList() {  
    struct DoublyLinkedList* list = (struct DoublyLinkedList*)malloc(sizeof(struct  
DoublyLinkedList));  
    if (!list) {  
        perror("Memory allocation failed");  
        exit(EXIT_FAILURE);  
    }  
    list->head = NULL;  
    list->tail = NULL;  
    return list;  
}
```

```
void append(struct DoublyLinkedList* list, int data) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    if (!newNode) {  
        perror("Memory allocation failed");  
        exit(EXIT_FAILURE);  
    }  
    newNode->data = data;  
    newNode->next = NULL;
```

```
    if (!list->head) {  
        newNode->prev = NULL;  
        list->head = newNode;  
        list->tail = newNode;  
    } else {  
        newNode->prev = list->tail;  
        list->tail->next = newNode;  
        list->tail = newNode;  
    }  
}
```

```
void displayList(struct DoublyLinkedList* list) {
    struct Node* current = list->head;
    int nodeNumber = 1;
    while (current) {
        printf(" node %d : %d\n", nodeNumber, current->data);
        current = current->next;
        nodeNumber++;
    }
}
```

```
void deleteNode(struct DoublyLinkedList* list, int position) {
    if (!list->head) {
        return;
    }
```

```
    struct Node* current = list->head;
    int count = 1;
```

```
    if (position == 1) {
        if (list->head->next) {
            list->head = list->head->next;
            list->head->prev = NULL;
        } else {
            list->head = NULL;
            list->tail = NULL;
        }
        free(current);
        return;
    }
```

```
    while (current && count < position) {
        current = current->next;
        count++;
    }
```

```
    if (!current) {
        printf("Invalid position. Try again.\n");
        return;
    }
```

```
    if (current->next) {
```

```

        current->prev->next = current->next;
        current->next->prev = current->prev;
    } else {
        list->tail = current->prev;
        list->tail->next = NULL;
    }
    free(current);
}

```

```

void freeList(struct DoublyLinkedList* list) {
    struct Node* current = list->head;
    struct Node* nextNode;
    while (current) {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }
    free(list);
}

```

```

int main() {
    int n, p;

    if (scanf("%d", &n) != 1) {
        fprintf(stderr, "Error reading the number of items.\n");
        return 1;
    }
}

```

```

    struct DoublyLinkedList* inventoryList = initializeList();

```

```

    if (n > 0) {
        for (int i = 0; i < n; i++) {
            int itemID;
            if (scanf("%d", &itemID) != 1) {
                fprintf(stderr, "Error reading item ID.\n");
                freeList(inventoryList);
                return 1;
            }
            append(inventoryList, itemID);
        }
    }
}

```

```
if (scanf("%d", &p) != 1) {  
    fprintf(stderr, "Error reading the position to delete.\n");  
    freeList(inventoryList);  
    return 1;  
}  
printf("Data entered in the list:\n");  
displayList(inventoryList);  
  
deleteNode(inventoryList, p);  
  
if (p >= 1 && p <= n)  
{  
    printf("After deletion the new list:\n");  
    if (inventoryList->head == NULL)  
    {  
    }  
    else  
    {  
        displayList(inventoryList);  
    }  
}  
  
freeList(inventoryList);  
  
return 0;  
}
```

Status : Correct

Marks : 10/10