

Rajalakshmi Engineering College

Name: madhumitha J

Email: 241501102@rajalakshmi.edu.in

Roll no: 241501102

Phone: 7904070243

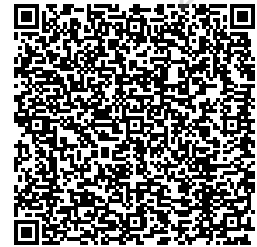
Branch: REC

Department: I AI & ML FA

Batch: 2028

Degree: B.E - AI & ML

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 1

Total Mark : 20

Marks Obtained : 15

Section 1 : MCQ

1. Where Fwd and Bwd represent forward and backward links to the adjacent elements of the list. Which of the following segments of code deletes the node pointed to by X from the doubly linked list, if it is assumed that X points to neither the first nor the last node of the list?

A doubly linked list is declared as

```
struct Node {  
    int Value;  
    struct Node *Fwd;  
    struct Node *Bwd;  
};
```

Answer

X->Bwd.Fwd = X->Fwd; X.Fwd->Bwd = X->Bwd;

Status : Wrong

Marks : 0/1

2. What will be the output of the following code?

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

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

int main() {
    struct Node* head = NULL;
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = 2;
    temp->next = NULL;
    temp->prev = NULL;
    head = temp;
    printf("%d\n", head->data);
    free(temp);
    return 0;
}
```

Answer

2

Status : Correct

Marks : 1/1

3. What is the main advantage of a two-way linked list over a one-way linked list?

Answer

Two-way linked lists allow for traversal in both directions.

Status : Correct

Marks : 1/1

4. Which of the following is false about a doubly linked list?

Answer

The insertion and deletion of a node take a bit longer

Status : Wrong

Marks : 0/1

5. Consider the following function that refers to the head of a Doubly Linked List as the parameter. Assume that a node of a doubly linked list has the previous pointer as prev and the next pointer as next.

Assume that the reference of the head of the following doubly linked list is passed to the below function 1 <--> 2 <--> 3 <--> 4 <--> 5 <--> 6. What should be the modified linked list after the function call?

Procedure fun(head_ref: Pointer to Pointer of node)

temp = NULL

current = *head_ref

While current is not NULL

temp = current->prev

current->prev = current->next

current->next = temp

current = current->prev

End While

If temp is not NULL

*head_ref = temp->prev

End If

End Procedure

Answer

6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1.

Status : Correct

Marks : 1/1

6. Which pointer helps in traversing a doubly linked list in reverse order?

Answer

prev

Status : Correct

Marks : 1/1

7. What will be the effect of setting the prev pointer of a node to NULL in a doubly linked list?

Answer

The node will become the new head

Status : Correct

Marks : 1/1

8. Which of the following is true about the last node in a doubly linked list?

Answer

Both prev and next pointers are NULL

Status : Wrong

Marks : 0/1

9. Consider the provided pseudo code. How can you initialize an empty two-way linked list?

Define Structure Node

data: Integer

prev: Pointer to Node

next: Pointer to Node

End Define

Define Structure TwoWayLinkedList

head: Pointer to Node

tail: Pointer to Node

End Define

Answer

```
struct TwoWayLinkedList* list = malloc(sizeof(struct TwoWayLinkedList)); list->head = NULL; list->tail = NULL;
```

Status : Correct

Marks : 1/1

10. Which code snippet correctly deletes a node with a given value from a doubly linked list?

```
void deleteNode(Node** head_ref, Node* del_node) {  
    if (*head_ref == NULL || del_node == NULL) {  
        return;  
    }  
    if (*head_ref == del_node) {  
        *head_ref = del_node->next;  
    }  
    if (del_node->next != NULL) {  
        del_node->next->prev = del_node->prev;  
    }  
    if (del_node->prev != NULL) {  
        del_node->prev->next = del_node->next;  
    }  
    free(del_node);  
}
```

Answer

Deletes the node with the highest data value in a doubly linked list.

Status : Wrong

Marks : 0/1

11. How do you delete a node from the middle of a doubly linked list?

Answer

All of the mentioned options

Status : Correct

Marks : 1/1

12. What is the correct way to add a node at the beginning of a doubly linked list?

Answer

```
void addFirst(int data){ Node* newNode = new Node(data);  newNode->  
&gt;next = head;      if (head != NULL) {      head-&gt;prev =  
newNode;  }  head = newNode;      }
```

Status : Correct

Marks : 1/1

13. How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

Status : Correct

Marks : 1/1

14. Which of the following information is stored in a doubly-linked list's nodes?

Answer

All of the mentioned options

Status : Correct

Marks : 1/1

15. How many pointers does a node in a doubly linked list have?

Answer

2

Status : Correct

Marks : 1/1

16. What is a memory-efficient double-linked list?

Answer

Each node has only one pointer to traverse the list back and forth

Status : Wrong

Marks : 0/1

17. What does the following code snippet do?

```
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
newNode->data = value;  
newNode->next = NULL;
```

```
newNode->prev = NULL;
```

Answer

Creates a new node and initializes its data to 'value'

Status : Correct

Marks : 1/1

18. Which of the following statements correctly creates a new node for a doubly linked list?

Answer

```
struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
```

Status : Correct

Marks : 1/1

19. What happens if we insert a node at the beginning of a doubly linked list?

Answer

The previous pointer of the new node is NULL

Status : Correct

Marks : 1/1

20. What will be the output of the following program?

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

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

```
int main() {
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < 5; i++) {
```

```
struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
temp->data = i + 1;
temp->prev = tail;
temp->next = NULL;
if (tail != NULL) {
    tail->next = temp;
} else {
    head = temp;
}
tail = temp;
}
struct Node* current = head;
while (current != NULL) {
    printf("%d ", current->data);
    current = current->next;
}
return 0;
}
```

Answer

1 2 3 4 5

Status : Correct

Marks : 1/1

Rajalakshmi Engineering College

Name: madhumitha J

Email: 241501102@rajalakshmi.edu.in

Roll no: 241501102

Phone: 7904070243

Branch: REC

Department: I AI & ML FA

Batch: 2028

Degree: B.E - AI & ML

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 2

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

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

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

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

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

```
struct Node {  
    char data;  
    Node* prev;  
    Node* next;
```

```
Node(char val) {  
    data = val;
```

```
    prev = NULL;  
    next = NULL;  
  }  
};
```

```
class DoublyLinkedList {  
public:
```

```
    Node* head;  
    Node* tail;
```

```
    DoublyLinkedList() {  
        head = NULL;  
        tail = NULL;  
    }
```

```
    void insertEnd(char val) {  
        Node* newNode = new Node(val);  
        if (!head) {  
            head = tail = newNode;  
        } else {  
            tail->next = newNode;  
            newNode->prev = tail;  
            tail = newNode;  
        }  
    }
```

```
    void insertFront(char val) {  
        Node* newNode = new Node(val);  
        if (!head) {  
            head = tail = newNode;  
        } else {  
            newNode->next = head;  
            head->prev = newNode;  
            head = newNode;  
        }  
    }
```

```
    void display() {  
        Node* curr = head;  
        while (curr) {  
            cout << curr->data;  
            if (curr->next) cout << " ";  
        }
```

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

```
int main() {
    DoublyLinkedList forwardList;
    DoublyLinkedList backwardList;
    char ch;
```

```
    while (cin >> ch && ch != '-') {
        forwardList.insertEnd(ch);
        backwardList.insertFront(ch);
    }
```

```
    cout << "Forward Playlist: ";
    forwardList.display();
```

```
    cout << "Backward Playlist: ";
    backwardList.display();
```

```
    return 0;
}
```

```
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 : Wrong

Marks : 0/10

Rajalakshmi Engineering College

Name: madhumitha J

Email: 241501102@rajalakshmi.edu.in

Roll no: 241501102

Phone: 7904070243

Branch: REC

Department: I AI & ML FA

Batch: 2028

Degree: B.E - AI & ML

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

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

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

```
void insert(struct Node **head, int id) {
    struct Node *new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->id = id;
    new_node->prev = NULL;
    new_node->next = *head;
```

```
    if (*head != NULL) {
        (*head)->prev = new_node;
    }
```

```
    *head = new_node;
}
```

```
void print_maximum(struct Node *head) {
    if (head == NULL) {
        printf("Empty list!\n");
```

```

    return;
}

struct Node *current = head;
int max_id = current->id;

while (current != NULL) {
    if (current->id > max_id) {
        max_id = current->id;
    }
    current = current->next;
}

printf("%d\n", max_id);
}

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

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

    struct Node *head = NULL;
    int id;
    for (int i = 0; i < n; i++) {
        scanf("%d", &id);
        insert(&head, id);
    }

    print_maximum(head);

    return 0;
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: madhumitha J

Email: 241501102@rajalakshmi.edu.in

Roll no: 241501102

Phone: 7904070243

Branch: REC

Department: I AI & ML FA

Batch: 2028

Degree: B.E - AI & ML

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* blink,*flink;
```

```
};
```

```
void insertatend(struct node** head,int x)
```

```
{
```

```
    struct node* newnode;
```

```
    newnode=(struct node*)malloc(sizeof(struct node));
```

```
    if (newnode!=NULL)
```

```
    {
```

```
        if (*head==NULL)
```

```
        {
```

```
            newnode->data=x;
```

```
            newnode->blink=NULL;
```

```
            newnode->flink=NULL;
```

```
            *head=newnode;
```

```
        }
```

```
    } else
```

```
    {
```

```
        struct node*p;
```

```

        p=*head;
        while (p->flink !=NULL)
            p=p->flink;
        newnode->data=x;
        newnode->flink=NULL;
        p->flink=newnode;
        newnode->blink=p;
    }
}
}
int main()
{
    struct node* head=NULL;
    int n,x;
    scanf("%d",&n);
    for (int i=0;i<n;i++)
    {
        scanf("%d",&x);
        insertatend(&head,x);
    }
    struct node *p=head;
    while (p!=NULL){
        printf("%d ",p->data);
        p=p->flink;
    }
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: madhumitha J

Email: 241501102@rajalakshmi.edu.in

Roll no: 241501102

Phone: 7904070243

Branch: REC

Department: I AI & ML FA

Batch: 2028

Degree: B.E - AI & ML

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
    struct node*p;
    p=start;
    printf("Node Inserted\n");
    while (p!=NULL)
    {
        printf("%d ",p->info);
```

```

        p=p->next;
    }
    printf("\n");
}

void insertAtFront(int data) {
    //type your code here
    struct node *newnode;
    newnode=(struct node*)malloc(sizeof(struct node));
    if (newnode!=NULL)
    {
        newnode->info=data;
        newnode->prev=NULL;
        newnode->next=start;
        if (start!=NULL)
            start=start->prev;
        start=newnode;
    }
}

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: madhumitha J

Email: 241501102@rajalakshmi.edu.in

Roll no: 241501102

Phone: 7904070243

Branch: REC

Department: I AI & ML FA

Batch: 2028

Degree: B.E - AI & ML

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 id;  
    struct node *prev;  
    struct node *next;  
};
```

```
struct node *head = NULL;  
struct node *tail = NULL;
```

```
void createList(int n) {  
    int i, id;  
    struct node *newNode;
```

```
    for (i = 0; i < n; i++) {  
        scanf("%d", &id);  
        newNode = (struct node *)malloc(sizeof(struct node));  
        newNode->id = id;  
        newNode->next = NULL;  
        newNode->prev = tail;
```

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

```
    }  
}
```

```
void deleteNode(int position) {  
    struct node *current = head;  
    int i;
```

```
    if (position < 1 || position > 20) {  
        printf("Invalid position. Try again.\n");  
        return;  
    }
```

```
    for (i = 1; current != NULL && i < position; i++) {  
        current = current->next;  
    }
```

```

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

    if (current->prev != NULL) {
        current->prev->next = current->next;
    } else {
        head = current->next;
    }

    if (current->next != NULL) {
        current->next->prev = current->prev;
    } else {
        tail = current->prev;
    }

    free(current);
}

void displayList() {
    struct node *current = head;
    int count = 1;

    printf("Data entered in the list:\n");
    while (current != NULL) {
        printf(" node %d : %d\n", count, current->id);
        count++;
        current = current->next;
    }
}

void displayAfterDeletion() {
    struct node *current = head;
    int count = 1;

    printf("\nAfter deletion the new list:\n");
    while (current != NULL) {
        printf(" node %d : %d\n", count, current->id);
        count++;
        current = current->next;
    }
}

```

```
}
```

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

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

```
    displayList();
```

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

```
    deleteNode(p);
```

```
    if (p >= 1 && p <= n) {  
        displayAfterDeletion();  
    }
```

```
    return 0;
```

```
}
```

Status : Correct

Marks : 10/10