

Rajalakshmi Engineering College

Name: Harish Suresh
Email: 241901034@rajalakshmi.edu.in
Roll no: 241901034
Phone: 7550029270
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 2

Section 1 : MCQ

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

Answer

2

Status : Correct

Marks : 1/1

2. 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

Status : Skipped

Marks : 0/1

3. How do you reverse a doubly linked list?

Answer

Status : Skipped

Marks : 0/1

4. 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

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

Answer

Status : Skipped

Marks : 0/1

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

Answer

Status : Skipped

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

8. 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

-

Status : -

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

11. 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

-

Status : -

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

14. 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

-

Status : -

Marks : 0/1

15. 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

-

Status : -

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

18. 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

-

Status : -

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

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

Answer

-

Status : -

Marks : 0/1

Rajalakshmi Engineering College

Name: Harish Suresh

Email: 241901034@rajalakshmi.edu.in

Roll no: 241901034

Phone: 7550029270

Branch: REC

Department: I CSE (CS) FA

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 : 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* last = *head;
    while (last->next != NULL) {
        last = last->next;
    }
    last->next = newNode;
    newNode->prev = last;
}

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

void displayBackward(struct Node* tail) {
    struct Node* current = tail;
    if (current == NULL) return;
    while (current->next != NULL) {
        current = current->next;
    }

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

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

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: Harish Suresh
Email: 241901034@rajalakshmi.edu.in
Roll no: 241901034
Phone: 7550029270
Branch: REC
Department: I CSE (CS) FA
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

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

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

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

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

```
struct DoublyLinkedList* createList() {  
    struct DoublyLinkedList* list = (struct DoublyLinkedList*)malloc(sizeof(struct  
DoublyLinkedList));  
    list->head = NULL;  
    list->tail = NULL;  
    return list;  
}
```

```
void append(struct DoublyLinkedList* list, int id) {
```

```

struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->id = id;
newNode->next = NULL;
if (list->head == NULL) {
    newNode->prev = NULL;
    list->head = newNode;
    list->tail = newNode;
    return;
}
newNode->prev = list->tail;
list->tail->next = newNode;
list->tail = newNode;
}

```

```

int printMaxID(struct DoublyLinkedList* list) {
    if (list->head == NULL) {
        return -1;
    }
    int maxID = list->head->id;
    struct Node* current = list->head;
    while (current != NULL) {
        if (current->id > maxID) {
            maxID = current->id;
        }
        current = current->next;
    }
    return maxID;
}

```

```

int main() {
    int n;
    scanf("%d", &n);
    struct DoublyLinkedList* list = createList();
    for (int i = 0; i < n; i++) {
        int id;
        scanf("%d", &id);
        append(list, id);
    }
    int maxID = printMaxID(list);
    if (maxID == -1) {
        printf("Empty list!\n");
    } else {

```

```
        printf("%d\n", maxID);  
    }  
    return 0;  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Harish Suresh
Email: 241901034@rajalakshmi.edu.in
Roll no: 241901034
Phone: 7550029270
Branch: REC
Department: I CSE (CS) FA
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 : 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;
```

```
void traverse() {
```

```
    struct node* current = start;
```

```
    while(current!=NULL) {
        printf("%d \n",current->info);
        current=current->next;
```

```
    }
}
```

```
void insertAtFront(int data) {
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->info = data;
    newNode->next = start;
    newNode->prev = NULL;
    if(start!=NULL) start->prev = newNode;
    start = 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: Harish Suresh
Email: 241901034@rajalakshmi.edu.in
Roll no: 241901034
Phone: 7550029270
Branch: REC
Department: I CSE (CS) FA
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

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

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

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

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

```
struct DoublyLinkedList* createList() {  
    struct DoublyLinkedList* list = (struct DoublyLinkedList*)malloc(sizeof(struct  
DoublyLinkedList));  
    list->head = NULL;  
    list->tail = NULL;  
    return list;  
}
```

```

void insert(struct DoublyLinkedList* list, int id) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->id = id;
    newNode->next = NULL;
    if (list->head == NULL) {
        newNode->prev = NULL;
        list->head = newNode;
        list->tail = newNode;
        return;
    }
    newNode->prev = list->tail;
    list->tail->next = newNode;
    list->tail = newNode;
}

void display(struct DoublyLinkedList* list) {
    struct Node* current = list->head;
    while (current != NULL) {
        printf("%d ", current->id);
        current = current->next;
    }
}

int main() {
    int N;
    scanf("%d", &N);
    struct DoublyLinkedList* list = createList();
    for (int i = 0; i < N; i++) {
        int id;
        scanf("%d", &id);
        insert(list, id);
    }
    display(list);
    printf("\n");
    return 0;
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Harish Suresh

Email: 241901034@rajalakshmi.edu.in

Roll no: 241901034

Phone: 7550029270

Branch: REC

Department: I CSE (CS) FA

Batch: 2028

Degree: B.E - CSE (CS)

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

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

```
struct DoublyLinkedList* createList() {
    struct DoublyLinkedList* list = (struct DoublyLinkedList*)malloc(sizeof(struct
DoublyLinkedList));
    list->head = NULL;
    list->tail = NULL;
    return list;
}
```

```
void insert(struct DoublyLinkedList* list, int id) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->id = id;
    newNode->next = NULL;
    if (list->head == NULL) {
        newNode->prev = NULL;
        list->head = newNode;
        list->tail = newNode;
        return;
    }
    newNode->prev = list->tail;
    list->tail->next = newNode;
    list->tail = newNode;
}
```

```
void display(struct DoublyLinkedList* list) {
    struct Node* current = list->head;
    int index = 1;
    while (current != NULL) {
        printf("node %d : %d\n", index, current->id);
        current = current->next;
    }
```

```

        index++;
    }
}

void deleteNode(struct DoublyLinkedList* list, int position) {
    if (position < 1) {
        printf("Invalid position. Try again.\n");
        return;
    }
    struct Node* current = list->head;
    for (int i = 1; i < position && current != NULL; i++) {
        current = current->next;
    }
    if (current == NULL) {
        printf("Invalid position. Try again.\n");
        return;
    }
    if (current->prev != NULL) {
        current->prev->next = current->next;
    } else {
        list->head = current->next;
    }
    if (current->next != NULL) {
        current->next->prev = current->prev;
    } else {
        list->tail = current->prev;
    }
    free(current);
}

int main() {
    int n, p;
    scanf("%d", &n);
    struct DoublyLinkedList* list = createList();
    for (int i = 0; i < n; i++) {
        int id;
        scanf("%d", &id);
        insert(list, id);
    }
    printf("Data entered in the list:\n");
    display(list);
    scanf("%d", &p);
}

```

```
deleteNode(list, p);  
if (p >= 1 && p <= n) {  
    printf("After deletion the new list:\n");  
    display(list);  
}  
return 0;  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Harish Suresh
Email: 241901034@rajalakshmi.edu.in
Roll no: 241901034
Phone: 7550029270
Branch: REC
Department: I CSE (CS) FA
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

Vanessa is learning about the doubly linked list data structure and is eager to play around with it. She decides to find out how the elements are inserted at the beginning and end of the list.

Help her implement a program for the same.

Input Format

The first line of input contains an integer N, representing the size of the doubly linked list.

The next line contains N space-separated integers, each representing the values to be inserted into the doubly linked list.

Output Format

The first line of output prints the integers, after inserting them at the beginning, separated by space.

The second line prints the integers, after inserting at the end, separated by space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

1 2 3 4 5

Output: 5 4 3 2 1

1 2 3 4 5

Answer

```
#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 insertAtBeginning(struct Node** head, int data) {  
    struct Node* newNode = createNode(data);  
    if (*head != NULL) {  
        (*head)->prev = newNode;  
        newNode->next = *head;
```

```

    }
    *head = newNode;
}

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

```

```

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

```

```

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

    int arr[N];
    for (i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    struct Node* headBegin = NULL;
    struct Node* headEnd = NULL;

    for (i = 0; i < N; i++) {
        insertAtBeginning(&headBegin, arr[i]);
    }
}

```

```
for (i = 0; i < N; i++) {  
    insertAtEnd(&headEnd, arr[i]);  
}  
  
printList(headBegin);  
printList(headEnd);  
  
return 0;  
}
```

Status : Correct

Marks : 10/10

2. Problem Statement

You are required to implement a program that deals with a doubly linked list.

The program should allow users to perform the following operations:

Insertion at the End: Insert a node with a given integer data at the end of the doubly linked list. Insertion at a given Position: Insert a node with a given integer data at a specified position within the doubly linked list. Display the List: Display the elements of the doubly linked list.

Input Format

The first line of input 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 elements to be inserted at the end.

The third line consists of integer m , representing the new element to be inserted.

The fourth line consists of an integer p , representing the position at which the new element should be inserted (1-based indexing).

Output Format

If p is valid, display the elements of the doubly linked list after performing the insertion at the specified position.

If p is invalid, display "Invalid position" in the first line and the second line prints

the original list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

10 25 34 48 57

35

4

Output: 10 25 34 35 48 57

Answer

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

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

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

```
Node* head = NULL;
```

```
void insertAtEnd(int data) {  
    Node* newNode = (Node*)malloc(sizeof(Node));  
    newNode->data = data;  
    newNode->next = NULL;  
    newNode->prev = NULL;
```

```
    if (head == NULL) {  
        head = newNode;  
        return;  
    }
```

```
    Node* temp = head;  
    while (temp->next != NULL)  
        temp = temp->next;
```

```
    temp->next = newNode;
```



```

    newNode->prev = temp;
}

int insertAtPosition(int data, int pos) {
    if (pos <= 0) return 0;

    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;

    if (pos == 1) {
        newNode->next = head;
        newNode->prev = NULL;
        if (head != NULL)
            head->prev = newNode;
        head = newNode;
        return 1;
    }

    Node* temp = head;
    int count = 1;
    while (temp != NULL && count < pos - 1) {
        temp = temp->next;
        count++;
    }

    if (temp == NULL || temp->next == NULL && count < pos - 1) return 0;

    newNode->next = temp->next;
    newNode->prev = temp;
    if (temp->next != NULL)
        temp->next->prev = newNode;
    temp->next = newNode;
    return 1;
}

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

```

```

    }
    printf("\n");
}

int main() {
    int n, m, p;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        int x;
        scanf("%d", &x);
        insertAtEnd(x);
    }

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

    if (!insertAtPosition(m, p)) {
        printf("Invalid position\n");
        display(); // print original list
    } else {
        display(); // print modified list
    }

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

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

```
}
```

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

```
void printReverseOrder(Node* tail) {  
    Node* current = tail;  
    while (current != NULL) {  
        printf("%d ", current->data);  
        current = current->prev;  
    }  
    printf("\n");  
}
```

```
int main() {  
    int n;  
  
    scanf("%d", &n);  
  
    if (n < 1 || n > 30) {  
        return 1;  
    }  
  
    Node* head = NULL;  
    Node* tail = NULL;  
  
    for (int i = 0; i < n; i++) {  
        int value;  
        scanf("%d", &value);  
        Node* newNode = createNode(value);  
        if (head == NULL) {  
            head = newNode;  
            tail = newNode;  
        } else {  
            tail->next = newNode;
```

```
        newNode->prev = tail;
        tail = newNode;
    }
}

printf("List in original order:\n");
printOriginalOrder(head);
printf("List in reverse order:\n");
printReverseOrder(tail);

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

return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Harish Suresh

Email: 241901034@rajalakshmi.edu.in

Roll no: 241901034

Phone: 7550029270

Branch: REC

Department: I CSE (CS) FA

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

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>
```

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

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

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

```
void printMiddle(Node* head, int n) {  
    Node* current = head;
```

```
    while (current != NULL) {  
        printf("%d ", current->data);  
        current = current->next;  
    }  
    printf("\n");
```

```
    if (n % 2 == 1) {  
        int midIndex = n / 2;  
        current = head;  
        for (int i = 0; i < midIndex; i++) {  
            current = current->next;  
        }  
        printf("%d\n", current->data);  
    } else {  
        int midIndex1 = (n / 2) - 1;  
        int midIndex2 = n / 2;  
        current = head;  
        for (int i = 0; i < midIndex1; i++) {  
            current = current->next;  
        }  
        printf("%d ", current->data);  
        current = current->next;  
        printf("%d\n", current->data);  
    }  
}
```

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

```
    Node* head = NULL;  
    Node* tail = NULL;
```

```
    for (int i = 0; i < n; i++) {  
        int data;  
        scanf("%d", &data);  
        Node* newNode = createNode(data);  
  
        if (head == NULL) {
```



```

        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }
}

printMiddle(head, n);

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

return 0;
}

```

Status : Correct

Marks : 10/10

2. 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* next;  
    struct Node* prev;  
} Node;
```

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

```
void insertEnd(Node** head, int data) {  
    Node* newNode = createNode(data);  
    if (*head == NULL) {
```

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

```
int isPalindrome(Node* head) {
    if (head == NULL) return 1;
    Node* left = head;
    Node* right = head;

    while (right->next != NULL) {
        right = right->next;
    }

    while (left != right && left->prev != right) {
        if (left->data != right->data) {
            return 0;
        }
        left = left->next;
        right = right->prev;
    }

    return 1;
}
```

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

```
int main() {
```

```

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

Node* head = NULL;

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

printList(head);

if (isPalindrome(head)) {
    printf("The doubly linked list is a palindrome\n");
} else {
    printf("The doubly linked list is not a palindrome\n");
}

return 0;
}

```

Status : Correct

Marks : 10/10

3. 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>
```

```
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 insertAtFront(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *head;
    if (*head != NULL) {
        (*head)->prev = newNode;
    }
    *head = newNode;
}
```

```
void insertAtPosition(struct Node** head, int position, int data) {
    if (position <= 0) return;
    struct Node* newNode = createNode(data);

    if (position == 1) {
        insertAtFront(head, data);
        return;
    }
```

```
    struct Node* current = *head;
    int count = 1;
```

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

```
    if (current == NULL) {
        printf("Position is out of bounds.\n");
        free(newNode);
        return;
    }
```

```
    newNode->next = current->next;
    current->next = newNode;
    newNode->prev = current;
```

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

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

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

```
    struct Node* head = NULL;
```

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

```
    printList(head);
```

```
    int position, data;  
    scanf("%d", &position);  
    scanf("%d", &data);  
    insertAtPosition(&head, position, data);
```

```
    printList(head);
```

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

```
}  
    return 0;  
}
```

Status : Correct

Marks : 10/10

4. 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>
```

```
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 appendFront(struct Node** head, int data) {  
    struct Node* newNode = createNode(data);  
    newNode->next = *head;  
    if (*head != NULL) {  
        (*head)->prev = newNode;  
    }  
    *head = newNode;  
}
```

```
void deleteNode(struct Node** head, int position) {  
    if (*head == NULL || position < 1) return;
```

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

```

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

    free(current);
}

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

int main() {
    int N, X;
    struct Node* head = NULL;

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

    scanf("%d", &X);

    printList(head);

    deleteNode(&head, X);

    // Output updated list
    printList(head);

```

```
    return 0;  
}
```

Status : Correct

Marks : 10/10

5. 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>  
  
#include <stdlib.h>
```

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

```
void append(struct Node** head_ref, int new_data) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    struct Node* last = *head_ref;
```

```
    new_node->data = new_data;
    new_node->next = NULL;
```

```
    if (*head_ref == NULL) {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }
```

```
    while (last->next != NULL)
        last = last->next;
```

```
    last->next = new_node;
    new_node->prev = last;
```

```
}
```

```
void rotate(struct Node** head_ref, int k) {
    if (k == 0 || *head_ref == NULL)
        return;
```

```
    struct Node* current = *head_ref;
    int count = 1;
```

```
    // Find the length of the list
    while (current->next != NULL) {
        current = current->next;
        count++;
    }
```

```
    k = k % count;
```

```
if (k == 0)
    return;
```

```
current->next = *head_ref;
(*head_ref)->prev = current;
```

```
int steps_to_new_head = count - k;
current = *head_ref;
for (int i = 0; i < steps_to_new_head - 1; i++)
    current = current->next;
```

```
*head_ref = current->next;
current->next->prev = NULL;
current->next = NULL;
```

```
}

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

```
int main() {
    struct Node* head = NULL;
    int n, k;
```

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

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

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

```
    rotate(&head, k);
```

```
    // Print the rotated list
    printList(head);
```

```
} return 0;
```

Status : Correct

Marks : 10/10