Rajalakshmi Engineering College

Name: Parameswari P
Email: 240701378@rajalakshmi.edu.in
Roll no: 240701378
Phone: 9500133836
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

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

Imagine Anu is tasked with finding the middle element of a doubly linked list. Given a doubly linked list where each node contains an integer value and is inserted at the end, implement a program to find the middle element of the list. If the number of nodes is even, return the middle element pair.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the doubly linked list.

The second line consists of N space-separated integers, representing the values of the nodes in the doubly linked list.

Output Format

The first line of output prints the space-separated elements of the doubly linked list.

The second line prints the middle element(s) of the doubly linked list, depending on whether the number of nodes is odd or even.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 10 20 30 40 50
30

Answer

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


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


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


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

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


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

void printMiddle(Node* head, int n) {
    Node* temp = head;
    int midIndex = n / 2;

    for (int i = 0; i < midIndex; i++) {
        temp = temp->next;
    }
    if (n % 2 == 1) {
        printf("%d\n", temp->data);
    } else {
        printf("%d %d\n", temp->prev->data, temp->data);
    }
}

int main() {
    int n, val;
    Node* head = NULL;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        insertEnd(&head, val);
    }
    printList(head);
    printMiddle(head, n);
```

```
    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

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

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

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

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

int length(Node* head) {
    int count = 0;
```

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

int insertAtPosition(Node** head, int val, int pos) {
    int len = length(*head);
    if (pos < 1 || pos > len + 1) {
        return 0;
    }

    Node* newNode = createNode(val);

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

    Node* temp = *head;
    for (int i = 1; i < pos - 1; i++) {
        temp = temp->next;
    }

    newNode->next = temp->next;
    newNode->prev = temp;

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

    return 1;
}

void display(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
```

```
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int n, val, pos;
    Node* head = NULL;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        insertEnd(&head, val);
    }
    scanf("%d", &val);
    scanf("%d", &pos);

    if (!insertAtPosition(&head, val, pos)) {
        printf("Invalid position\n");
        display(head);
    } else {
        display(head);
    }

    return 0;
}
```

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

3. Problem Statement

Ashiq is developing a ticketing system for a small amusement park. The park issues tickets to visitors in the order they arrive. However, due to a system change, the oldest ticket (first inserted) must be revoked instead of the last one.

To manage this, Ashiq decided to use a doubly linked list-based stack, where:

Pushing adds a new ticket to the top of the stack.Removing the first inserted ticket (removing from the bottom of the stack).Printing the

remaining tickets from bottom to top.

Input Format

The first line consists of an integer n, representing the number of tickets issued.

The second line consists of n space-separated integers, each representing a ticket number in the order they were issued.

Output Format

The output prints space-separated integers, representing the remaining ticket numbers in the order from bottom to top.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
24 96 41 85 97 91 13

Output: 96 41 85 97 91 13

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

void push(Node** top, int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;
    newNode->prev = *top;
    if (*top != NULL) {
        (*top)->next = newNode;
    }
    *top = newNode;
}
```

```
void removeBottom(Node** top) {
    if (*top == NULL) return;
    Node* temp = *top;
    while (temp->prev != NULL) {
        temp = temp->prev;
    }
    if (temp->next == NULL) {
        *top = NULL;
    } else {
        temp->next->prev = NULL;
    }

    free(temp);
}

void printBottomToTop(Node* top) {
    if (top == NULL) return;

    Node* temp = top;
    while (temp->prev != NULL) {
        temp = temp->prev;
    }
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int n, ticket;
    Node* top = NULL;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &ticket);
        push(&top, ticket);
    }
    removeBottom(&top);
    printBottomToTop(top);

    return 0;
}
```

Status : Correct *Marks : 10/10*