Rajalakshmi Engineering College

Name: NITHISH RAJ L
Email: 240701366@rajalakshmi.edu.in
Roll no: 2116240701366
Phone: 8072719523
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

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) {
    printf("List in original order:\n");
    Node* current = head;
    while (current != NULL) {
```

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

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

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

    int value;
    Node* head = NULL;
    Node* tail = NULL;

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

    printOriginalOrder(head);
    printReverseOrder(tail);

    return 0;
}
```

Status : Correct *Marks : 10/10*

2. Problem Statement

Aarav is working on a program to analyze his test scores, which are stored in a doubly linked list. He needs a solution to input scores into the list and determine the highest score.

Help him by providing code that lets users enter test scores into the doubly linked list and find the maximum score efficiently.

Input Format

The first line 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 score to be inserted.

Output Format

The output prints an integer, representing the highest score present in the list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
89 71 2 70
Output: 89

Answer

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

typedef struct Node {
```

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

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

void insertNode(Node** head, int score) {
    Node* newNode = createNode(score);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}

int findMaxScore(Node* head) {
    if (head == NULL) {
        return INT_MIN;
    }
    int maxScore = head->score;
    Node* temp = head->next;
    while (temp != NULL) {
        if (temp->score > maxScore) {
            maxScore = temp->score;
        }
        temp = temp->next;
    }
    return maxScore;
}
```

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

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

    Node* head = NULL;

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

    printf("%d\n", findMaxScore(head));

    freeList(head);

    return 0;
}
```

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

3. 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 new_data) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->prev = NULL;
    new_node->next = NULL;
    return new_node;
}
```

```
void insertAtBeginning(struct Node** head, int new_data) {
    struct Node* new_node = createNode(new_data);
    new_node->next = *head;
    if (*head != NULL) {
        (*head)->prev = new_node;
    }
    *head = new_node;
}

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

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

void printReverse(struct Node* head) {
    if (head == NULL) return;

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

    while (temp != NULL) {
        printf("%d ", temp->data);
```

```
        temp = temp->prev;
    }
    printf("\n");
}

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

    int value;
    struct Node* head = NULL;

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

    printList(head);

    struct Node* tail = NULL;
    struct Node* temp = head;
    while (temp != NULL) {
        insertAtEnd(&tail, temp->data);
        temp = temp->next;
    }

    printReverse(tail);

    return 0;
}
```

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