Rajalakshmi Engineering College

Name: Siddesh Kumar L
Email: 240701512@rajalakshmi.edu.in
Roll no: 240701512
Phone: null
Branch: REC
Department: I CSE FE
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

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>

// Define Node structure
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

Node* head = NULL;

// Append node to the end of the list
void append(int value) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = value;
    new_node->next = NULL;
    new_node->prev = NULL;

    if (head == NULL) {
        head = new_node;
        return;
    }

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

```
    temp->next = new_node;
    new_node->prev = temp;
}

// Function to find the maximum score
int find_max() {
    if (head == NULL)
        return -1; // return -1 if list is empty

    int max = head->data;
    Node* temp = head->next;
    while (temp != NULL) {
        if (temp->data > max)
            max = temp->data;
        temp = temp->next;
    }
    return max;
}

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

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

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

2. 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

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

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

void insert_at_beginning(Node** head, int value) {
    Node* new_node = (Node*)malloc(sizeof(Node));
```

```
    new_node->data = value;
    new_node->prev = NULL;
    new_node->next = *head;

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

    *head = new_node;
}

void insert_at_end(Node** head, int value) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = value;
    new_node->next = NULL;
    new_node->prev = NULL;

    if (*head == NULL) {
        *head = new_node;
        return;
    }

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

    temp->next = new_node;
    new_node->prev = temp;
}

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

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

```
    Node* head_begin = NULL;
    Node* head_end = NULL;

    for (int i = 0; i < N; i++) {
        scanf("%d", &value);
        insert_at_beginning(&head_begin, value);
        insert_at_end(&head_end, value);
    }

    print_list(head_begin);
    print_list(head_end);

    return 0;
}
```

Status : Correct *Marks : 10/10*

3. 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* next;
    struct Node* prev;
} Node;

Node* head = NULL;


void insert_at_end(int value) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = value;
    new_node->next = NULL;
    new_node->prev = NULL;

    if (head == NULL) {
        head = new_node;
        return;
    }

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

    temp->next = new_node;
    new_node->prev = temp;
}
```

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

void find_middle(int N) {
    Node* temp = head;
    int count = 0;

    for (int i = 0; i < N / 2; 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, value;
    scanf("%d", &N);

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

    print_list();
    find_middle(N);

    return 0;
}
```

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