Rajalakshmi Engineering College

Name: SHREYA KS
Email: 240701506@rajalakshmi.edu.in
Roll no: 240701506
Phone: 9789293683
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

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

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

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

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

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

    for(int i=0;i<mid;i++){
        temp=temp->next;
    }
    if(n % 2 == 0){
        printf("%d %d\n",temp->prev->data,temp->data);
    }else{
        printf("%d\n", temp->data);
    }
}
int main(){
    int n,value;
    Node* head =NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&value);
        insertEnd(&head,value);
    }

    printList(head);
    printMiddle(head,n);

    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>
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 insertEnd(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;
}
int findMax(Node* head){
    if (head == NULL) return -1;

    int max = head->data;
    struct Node*temp = head->next;

    while(temp != NULL) {
        if(temp->data > max) {
            max = temp->data;
        }
        temp = temp->next;
    }

    return max;

}
int main() {
    int n, value;
    struct Node* head = NULL;

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

```
    }

    int maximum = findMax(head);
    printf("%d\n", maximum);

    return 0;

}
```

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

3. 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>

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 insertEnd(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;
}

int insertAtPosition(struct Node** head, int pos, int data) {
    struct Node* newNode = createNode(data);
    if(pos <= 0) {
        return 0;
    }
    if (pos == 1) {
        newNode->next = *head;
        if(*head != NULL)
            (*head)->prev = newNode;
        *head = newNode;
         return 1;
    }
    struct Node* temp = *head;
    for(int i = 1; i < pos - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if(temp == NULL) {
        return 0;
    }
    newNode->next = temp->next;
    if (temp->next != NULL)
          temp->next->prev = newNode;
    temp->next = newNode;
    newNode->prev = temp;
    return 1;
    }


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

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

```
    int n, value, m, pos;

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

    scanf("%d", &pos);

    int success = insertAtPosition(&head, pos, m);
    if(!success){
        printf("Invalid position\n");
    }

    displayList(head);

    return 0;
}
```

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