Rajalakshmi Engineering College

Name: Vikashini M
Email: 241801314@rajalakshmi.edu.in
Roll no: 241801314
Phone: 9345747519
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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>

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;
}
void printList(struct Node* head){
    struct Node* temp = head;
    while(temp != NULL){
        printf("%d",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
void findMiddle(struct Node* head,int n){
    struct Node* slow=head;
    struct Node* fast=head;
    int count=0;
    while(fast != NULL && fast->next != NULL){
        slow = slow->next;
        fast=fast->next->next;
        count++;
    }
    if(n%2==0){
        printf("%d %d\n",slow->prev->data,slow->data);
    }else{
        printf("%d\n",slow->data);
    }
}
int main(){
    int n,data;
    struct Node* head=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&data);
        insertEnd(&head,data);
    }
    printList(head);
    findMiddle(head,n);
    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

```
#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 insertBegin(struct Node** head,int data){
    struct Node* newNode = createNode(data);
    if(*head == NULL){
        *head = newNode;
        return;
    }
    newNode->next= *head;
    (*head)->prev=newNode;
    *head=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;
}
void printList(struct Node* head){
    struct Node*temp=head;
    while(temp != NULL){
        printf("%d ",temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main(){
    int n,data;
    struct Node* headBegin = NULL;
```

```
    struct Node* headEnd = NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&data);
        insertBegin(&headBegin,data);
        insertEnd(&headEnd,data);
    }
    printList(headBegin);
    printList(headEnd);
    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* 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;
}
int main(){
    int n,data;
    scanf("%d",&n);
    Node* head = NULL;
    Node* tail = NULL;
    for(int i=0;i<n;i++){
        scanf("%d",&data);
        Node* newNode = createNode(data);
        if(head == NULL){
            head = newNode;
            tail = newNode;
        }
        else{
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
```

```
        }
    }
    printf("List in original order:\n");
    Node* temp = head;
    while(temp != NULL){
        printf("%d ",temp->data);
        temp = temp->next;
    }
    printf("\n");
    printf("List in reverse order:\n");
    temp = tail;
    while(temp != NULL){
        printf("%d ",temp->data);
        temp=temp->prev;
    }
    printf("\n");
    return 0;
}
```

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