

Rajalakshmi Engineering College

Name: Sherin Katherina
Email: 240701495@rajalakshmi.edu.in
Roll no: 240701495
Phone: 9150930353
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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int element;
    struct node*Prev;
    struct node*Next;
};
typedef struct node Node;
int IsEmpty(Node*List){
    return(List->Next==NULL);
}
void InsertEnd(Node*List,int e){
    Node*Newnode=(Node*)malloc(sizeof(Node));
    Newnode->element=e;
    Newnode->Next=NULL;
    if(IsEmpty(List)){
        Newnode->Prev=List;
        List->Next=Newnode;
    }
    else{
        Node*Position=List->Next;
        while(Position->Next!=NULL){
            Position=Position->Next;
        }
        Position->Next=Newnode;
```

```

        Newnode->Prev=Position;
    }
}

void Traverse(Node*List){
    Node*Position=List->Next;
    while(Position!=NULL){
        printf("%d ",Position->element);
        Position=Position->Next;
    }
    printf("\n");
}

void findmid(Node*List){
    if(List->Next==NULL){
        printf("\n");
        return;
    }
    Node*slow=List->Next;
    Node*fast=List->Next;
    while(fast->Next!=NULL && fast->Next->Next!=NULL){
        slow=slow->Next;
        fast=fast->Next->Next;
    }
    if(fast->Next==NULL){
        printf("%d\n",slow->element);
    }
    else{
        printf("%d %d\n",slow->element,slow->Next->element);
    }
}

void Free(Node*List){
    Node*Position=List;
    while(Position!=NULL){
        Node*temp=Position;
        Position=Position->Next;
        free(temp);
    }
}

int main(){
    Node*List=(Node*)malloc(sizeof(Node));
    List->Prev=NULL;
    List->Next=NULL;

```

```
int n,e;
scanf("%d",&n);
for(int i=0;i<n;i++){
    scanf("%d",&e);
    InsertEnd(List,e);
}
Traverse(List);
findmid(List);
Free(List);
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>
```

```
typedef struct node{
```

```
    int element;
```

```
    struct node*next;
```

```
    struct node*prev;
```

```
}Node;
```

```
void insertbeg(Node*List1,int e){
```

```
    Node*newnode=(Node*)malloc(sizeof(Node));
```

```
    newnode->element=e;
```

```
    if(List1->next==NULL){
```

```
        newnode->next=NULL;
```

```
    }
```

```
    else{
```

```
        newnode->next=List1->next;
```

```
        List1->next->prev=newnode;
```

```
    }
```

```
    newnode->prev=List1;
```

```
    List1->next=newnode;
```

```
}
```

```
void insertend(Node*List2,int e){
```

```
    Node*newnode=(Node*)malloc(sizeof(Node));
```

```
    Node*Position;
```

```
    newnode->element=e;
```

```
    newnode->next=NULL;
```

```
    if(List2->next==NULL){
```

```
        newnode->prev=List2;
```

```
        List2->next=newnode;
```

```
    }
```

```
    else{
```

```
        Position=List2;
```

```
        while(Position->next!=NULL){
```

```
            Position=Position->next;
```

```

    }
    newnode->prev=Position;
    Position->next=newnode;
}
}
void traverse(Node*List){
    Node*Position=List->next;
    while(Position!=NULL){
        printf("%d ",Position->element);
        Position=Position->next;
    }
    printf("\n");
}
int main(){
    Node*List1=(Node*)malloc(sizeof(Node));
    Node*List2=(Node*)malloc(sizeof(Node));
    List1->next=NULL;
    List2->next=NULL;
    int n,e;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&e);
        insertbeg(List1,e);
        insertend(List2,e);
    }
    traverse(List1);
    traverse(List2);
    return 0;
}

```

Status : Correct

Marks : 10/10

3. Problem Statement

Krishna needs to create a doubly linked list to store and display a sequence of integers. Your task is to help write a program to read a list of integers from input, store them in a doubly linked list, and then display the list.

Input Format

The first line of input consists of an integer n, representing the number of integers in the list.

The second line of input consists of n space-separated integers.

Output Format

The output prints a single line displaying the integers in the order they were added to the doubly linked list, separated by spaces.

If nothing is added (i.e., the list is empty), it will display "List is empty".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

1 2 3 4 5

Output: 1 2 3 4 5

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int element;
    struct node*next;
    struct node*prev;
}Node;
void insertend(Node*List,int e){
    Node*newnode=(Node*)malloc(sizeof(Node));
    Node*position;
    newnode->next=NULL;
    newnode->element=e;
    if(List->next==NULL){
        newnode->prev=List;
        List->next=newnode;
    }
    else{
        position=List;
        while(position->next!=NULL){
```

```

        position=position->next;
    }
    newnode->prev=position;
    position->next=newnode;
}
}
void traverse(Node*List){
    if(List->next!=NULL){
        Node*Position=List->next;
        while(Position!=NULL){
            printf("%d ",Position->element);
            Position=Position->next;
        }
        printf("\n");
    }
    else{
        printf("List is empty\n");
    }
}
int main(){
    Node*List=(Node*)malloc(sizeof(Node));
    List->next=NULL;
    int n,e;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&e);
        insertend(List,e);
    }
    traverse(List);
    return 0;
}

```

Status : Correct

Marks : 10/10