Rajalakshmi Engineering College

Name: Shasmeen Syed
Email: 240701492@rajalakshmi.edu.in
Roll no: 2116240701492
Phone: 9677485510
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_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is developing a student registration system for a college. To efficiently store and manage the student IDs, he decides to implement a doubly linked list where each node represents a student's ID.

In this system, each student's ID is stored sequentially, and the system needs to display all registered student IDs in the order they were entered.

Implement a program that creates a doubly linked list, inserts student IDs, and displays them in the same order.

Input Format

The first line contains an integer N the number of student IDs.

The second line contains N space-separated integers representing the student IDs.

Output Format

The output should display the single line containing N space-separated integers representing the student IDs stored in the doubly linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
Output: 10 20 30 40 50

Answer

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

struct node{
   struct node*prev;
   int element;
   struct node*Next;
};
typedef struct node Node;

int IsEmpty(Node*List)
{
   if(List->Next==NULL)
   {
      return 1;
   }
   else{
      return 0;
   }
}

void InsertLast(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;
        while(Position->Next!=NULL)
        {
            Position=Position->Next;
        }
        Position->Next=NewNode;
        NewNode->prev=Position;
    }
}

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

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

```
    int e,N;
    scanf("%d",&N);
    for(int i=0;i<N;i++)
    {
        scanf("%d",&e);
        InsertLast(List,e);
    }
    Traverse(List);
    return 0;
}
```

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