Rajalakshmi Engineering College

Name: Sarvesh S
Email: 240701479@rajalakshmi.edu.in
Roll no: 240701479
Phone: 9361488694
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

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>
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**

2. Problem Statement

Ashiq is developing a ticketing system for a small amusement park. The park issues tickets to visitors in the order they arrive. However, due to a system change, the oldest ticket (first inserted) must be revoked instead of the last one.

To manage this, Ashiq decided to use a doubly linked list-based stack, where:

Pushing adds a new ticket to the top of the stack.Removing the first inserted ticket (removing from the bottom of the stack).Printing the remaining tickets from bottom to top.

Input Format

The first line consists of an integer n, representing the number of tickets issued.

The second line consists of n space-separated integers, each representing a ticket number in the order they were issued.

Output Format

The output prints space-separated integers, representing the remaining ticket numbers in the order from bottom to top.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
24 96 41 85 97 91 13

Output: 96 41 85 97 91 13

Answer

```
#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){
    Node *position=List->next;
    while(position!=NULL){
        printf("%d ",position->element);
        position=position->next;
    }
    printf("\n");
}

void deletebeg(Node *List){
    if(List->next!=NULL){
        Node *Tempnode=List->next;
        List->next=Tempnode->next;
        if(List->next!=NULL){
            Tempnode->next->prev=List;
            free(Tempnode);
        }
    }
```

```
}
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);
    }
    deletebeg(List);
    traverse(List);
    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

```
#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**