Rajalakshmi Engineering College

Name: ROHITH KUMAR S
Email: 240701436@rajalakshmi.edu.in
Roll no: 240701436
Phone: 7603815548
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

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

```c
#include<stdio.h>
#include<stdlib.h>
struct node{
    int e;
    struct node *prev;
    struct node *next;
};
typedef struct node Node;
int main(){
    Node *List=NULL;
    Node *pos;
    int n;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        Node *newnode=(Node*)malloc(sizeof(Node));
        scanf("%d",&newnode->e);
        newnode->next=NULL;
        newnode->prev=NULL;
        if(List==NULL){
            List=newnode;
        }else{
            pos=List;
            while(pos->next!=NULL){
                pos=pos->next;
            }
            pos->next=newnode;
            newnode->prev=pos;
```

```
      }
   }
   int max=List->e;
   pos=List;
   while(pos!=NULL){
      if(max<pos->e){
         max=pos->e;
      }
      pos=pos->next;
   }
   printf("%d",max);
}
```

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

2. Problem Statement

Imagine you're managing a store's inventory list, and some products were accidentally entered multiple times. You need to remove the duplicate products from the list to ensure each product appears only once.

You have an unsorted doubly linked list of product IDs. Some of these product IDs may appear more than once, and your goal is to remove any duplicates.

Input Format

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

The second line of input consists of n space-separated integers representing the list elements.

Output Format

The output prints the final after removing duplicate nodes, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
12 12 10 4 8 4 6 4 4 8
Output: 8 4 6 10 12

Answer

```c
#include<stdio.h>
#include<stdlib.h>
struct node{
    int e;
    struct node *next;
    struct node *prev;
};
typedef struct node Node;
int main(){
    Node *List=NULL;
    Node *pos;
    int n;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        Node*newnode=(Node*)malloc(sizeof(Node));
        scanf("%d",&newnode->e);
        newnode->next=NULL;
        newnode->prev=NULL;
        if(List==NULL){
            List=newnode;
        }else{
            pos=List;
            while(pos->next!=NULL){
                pos=pos->next;
            }
            pos->next=newnode;
            newnode->prev=pos;
        }
    }
    pos=List;
    while(pos->next!=NULL){
        pos=pos->next;
    }
    Node *temp;
    while(pos!=NULL){
```

```
    int k=pos->e;
    temp=pos->next;
    int v=0;
    while(temp!=NULL){
        if(temp->e==k){
            v=1;
            break;
        }
        temp=temp->next;
    }
    if(v==0){
        printf("%d ",pos->e);
    }
    pos=pos->prev;
  }
}
```

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

```c
#include<stdio.h>
#include<stdlib.h>
struct node{
    int e;
    struct node *next;
    struct node *prev;
};
typedef struct node Node;
int main(){
    Node *List=NULL;
    Node *pos;
    int n;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        Node *newnode=(Node*)malloc(sizeof(Node));
        scanf("%d",&newnode->e);
        newnode->next=NULL;
        newnode->prev=NULL;
        if(List==NULL){
            List=newnode;
```

```c
        }else{
            pos=List;
            while(pos->next!=NULL){
                pos=pos->next;
            }
            pos->next=newnode;
            newnode->prev=pos;
        }
    }
    int k;
    scanf("%d",&k);
    int f;
    scanf("%d",&f);
    int c=1;
    pos=List;
    while(pos!=NULL){
        c++;
        pos=pos->next;
    }
    if(f>c){
        printf("Invalid position\n");
        pos=List;
        while(pos!=NULL){
            printf("%d ",pos->e);
            pos=pos->next;
        }
    }else{
        int v=1;
        pos=List;
        while(pos->next!=NULL&&v<f-1){
            v++;
            pos=pos->next;
        }
        Node *tempnode=(Node*)malloc(sizeof(Node));
        tempnode->e=k;
        tempnode->prev=pos;
        tempnode->next=pos->next;
        if(pos->next!=NULL){
            pos->next->prev=tempnode;
        }
        pos->next=tempnode;
        pos=List;
```

```
while(pos!=NULL){
    printf("%d ",pos->e);
    pos=pos->next;
}


    }
}
```

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