NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 10

Section 1 : Coding

1. 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>

#define MAX_SIZE 15

struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

struct Stack {
    struct Node* top;
    struct Node* bottom;
};

void push(struct Stack* stack, int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;
    newNode->prev = stack->top;

    if (stack->top == NULL) {
```

```
        stack->top = newNode;
        stack->bottom = newNode;
    } else {
        stack->top->next = newNode;
        stack->top = newNode;
    }
}

void removeBottom(struct Stack* stack) {
    if (stack->bottom == NULL) {
        return;
    }
    struct Node* temp = stack->bottom;
    stack->bottom = stack->bottom->next;
    if (stack->bottom == NULL) {
        stack->top = NULL;
    } else {
        stack->bottom->prev = NULL;
    }
    free(temp);
}

void printBottomToTop(struct Stack* stack) {
    struct Node* current = stack->bottom;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
}

int main() {
    int n;
    scanf("%d", &n);

    if (n < 1 || n > MAX_SIZE) {
        return 1;
    }

    struct Stack stack = {NULL, NULL};

    for (int i = 0; i < n; i++) {
        int ticket;
```

```
        scanf("%d", &ticket);
        if (ticket < 1 || ticket > 100000) {
            return 1;
        }
        push(&stack, ticket);
    }

    removeBottom(&stack);

    printBottomToTop(&stack);

    struct Node* current = stack.bottom;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }

    return 0;
}
```

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

2. 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

-

Status : Skipped ***Marks : 0/10***

3. 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

-

Status : *-* ***Marks : 0/10***