Rajalakshmi Engineering College

Name: Nitin Aakash
Email: 240701370@rajalakshmi.edu.in
Roll no: 240701370
Phone: 9498349045
Branch: REC
Department: I CSE FD
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

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>


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


struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return newNode;
}

void push(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
```

```
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}


void popFirst(struct Node** head) {
    if (*head == NULL) return;

    struct Node* temp = *head;

    if (temp->next == NULL) {
        *head = NULL;
    } else {
        *head = temp->next;
        (*head)->prev = NULL;
    }

    free(temp);
}


void displayStack(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

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

    struct Node* stack = NULL;

    for (int i = 0; i < n; i++) {
        int ticket;
        scanf("%d", &ticket);
        push(&stack, ticket);
    }

    popFirst(&stack);


    displayStack(stack);
    printf("\n");

    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

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

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

struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return newNode;
}

void insertAtBeginning(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   newNode->next = *head;
   if (*head != NULL) {
      (*head)->prev = newNode;
   }
   *head = newNode;
}

void insertAtEnd(struct Node** head, int data) {
   struct Node* newNode = createNode(data);

   if (*head == NULL) {
      *head = newNode;
      return;
```

```
    }

    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

void displayList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

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

    struct Node* head = NULL;

    int values[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &values[i]);
    }

    for (int i = 0; i < N; i++) {
        insertAtBeginning(&head, values[i]);
    }

    displayList(head);

    struct Node* newHead = NULL;
    for (int i = 0; i < N; i++) {
        insertAtEnd(&newHead, values[i]);
    }

    displayList(newHead);
    return 0;
}
```

Status : Correct **_Marks : 10/10_**

3. 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

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

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);

    if (*head == NULL) {
        *head = newNode;
        return;
    }

    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }

    temp->next = newNode;
    newNode->prev = temp;
}

int findMax(struct Node* head) {
    if (head == NULL) return -1;

    int max = head->data;
    struct Node* temp = head->next;

    while (temp != NULL) {
        if (temp->data > max) {
            max = temp->data;
        }
        temp = temp->next;
    }

    return max;
}

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

    struct Node* head = NULL;

    for (int i = 0; i < N; i++) {
        int score;
        scanf("%d", &score);
        insertEnd(&head, score);
    }

    int maxScore = findMax(head);
    printf("%d\n", maxScore);

    return 0;
}
```

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