Rajalakshmi Engineering College

Name: Seeralan .M
Email: 241501193@rajalakshmi.edu.in
Roll no: 241501193
Phone: 8610861705
Branch: REC
Department: I AIML AE
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

A customer support system is designed to handle incoming requests using a queue. Implement a linked list-based queue where each request is represented by an integer. After processing the requests, remove any duplicate requests to ensure that each request is unique and print the remaining requests.

Input Format

The first line of input consists of an integer N, representing the number of requests to be enqueued.

The second line consists of N space-separated integers, each representing a request.

Output Format

The output prints space-separated integers after removing the duplicate requests.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 2 7 5

Output: 2 4 7 5

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

// Node structure for the queue
struct Node {
    int data;
    struct Node* next;
};

// Queue front and rear pointers
struct Node* front = NULL;
struct Node* rear = NULL;

// Function to enqueue a request
void enqueue(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (rear == NULL) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
}
```

```
// Function to print the queue and remove duplicates
void printUniqueRequests() {
    int seen[101] = {0}; // Request values are between 1 and 100
    struct Node* current = front;

    while (current != NULL) {
        if (!seen[current->data]) {
            printf("%d ", current->data);
            seen[current->data] = 1;
        }
        current = current->next;
    }
    printf("\n");
}

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

    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        enqueue(val);
    }

    printUniqueRequests();
    return 0;
}
```

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

2. Problem Statement

Manoj is learning data structures and practising queues using linked lists. His professor gave him a problem to solve. Manoj started solving the program but could not finish it. So, he is seeking your assistance in solving it.

The problem is as follows: Implement a queue with a function to find the Kth element from the end of the queue.

Help Manoj with the program.

Input Format

The first line of input consists of an integer N, representing the number of elements in the queue.

The second line consists of N space-separated integers, representing the queue elements.

The third line consists of an integer K.

Output Format

The output prints an integer representing the Kth element from the end of the queue.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 6 7 5
3

Output: 6

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

// Structure to represent a queue node
struct Node {
    int data;
    struct Node* next;
};

// Function to enqueue an element into the queue
void enqueue(struct Node** front, struct Node** rear, int value) {
    // Create a new node
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
```

```
    newNode->data = value;
    newNode->next = NULL;

    // If the queue is empty, the new node is both the front and rear
    if (*rear == NULL) {
        *front = *rear = newNode;
        return;
    }

    // Add the new node to the rear of the queue
    (*rear)->next = newNode;
    *rear = newNode;
}

// Function to find the Kth element from the end of the queue
int findKthFromEnd(struct Node* front, int K) {
    struct Node *first = front, *second = front;

    // Move the first pointer K steps ahead
    for (int i = 0; i < K; i++) {
        if (first == NULL) {
            return -1; // K is greater than the length of the queue
        }
        first = first->next;
    }

    // Now move both pointers one step at a time
    while (first != NULL) {
        first = first->next;
        second = second->next;
    }

    // The second pointer is now at the Kth element from the end
    return second->data;
}

int main() {
    struct Node* front = NULL;
    struct Node* rear = NULL;
    int N, K;

    // Read the number of elements in the queue
```

```
    scanf("%d", &N);

    // Read the elements and enqueue them
    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        enqueue(&front, &rear, value);
    }

    // Read the value of K
    scanf("%d", &K);

    // Find and print the Kth element from the end
    int result = findKthFromEnd(front, K);
    if (result != -1) {
        printf("%d\n", result);
    } else {
        printf("Invalid K\n");
    }

    return 0;
}
```

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

3. Problem Statement

Saran is developing a simulation for a theme park where people wait in a queue for a popular ride.

Each person has a unique ticket number, and he needs to manage the queue using a linked list implementation.

Your task is to write a program for Saran that reads the number of people in the queue and their respective ticket numbers, enqueue them, and then calculate the sum of all ticket numbers to determine the total ticket value present in the queue.

Input Format

The first line of input consists of an integer N, representing the number of people

in the queue.

The second line consists of N space-separated integers, representing the ticket numbers.

Output Format

The output prints an integer representing the sum of all ticket numbers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 6 7 5
Output: 24

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

// Define the structure of a queue node
struct Node {
    int data;
    struct Node* next;
};

// Define front and rear pointers
struct Node* front = NULL;
struct Node* rear = NULL;

// Function to enqueue a ticket number
void enqueue(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (rear == NULL) {
        front = rear = newNode;
    } else {
```

```
        rear->next = newNode;
        rear = newNode;
    }
}

// Function to calculate the sum of ticket numbers in the queue
int calculateSum() {
    int sum = 0;
    struct Node* temp = front;
    while (temp != NULL) {
        sum += temp->data;
        temp = temp->next;
    }
    return sum;
}

int main() {
    int n, val;
    scanf("%d", &n);  // Number of people/tickets

    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        enqueue(val);
    }

    int result = calculateSum();
    printf("%d\n", result);

    return 0;
}
```

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