Rajalakshmi Engineering College

Name: Piraisoodan R
Email: 240701384@rajalakshmi.edu.in
Roll no: 240701384
Phone: 8056892546
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 4_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Fathima has been tasked with developing a program to manage a queue of customers waiting in line at a service center. Help her write a program simulating a queue data structure using a linked list.

Here is a description of the scenario and the required operations:

Enqueue: Add a customer to the end of the queue.Dequeue: Remove and discard a customer from the front of the queue.Display waiting customers: Display the front and rear customer IDs in the queue.

Write a program that enqueues all the customers into the queue, performs a dequeue operation, and prints the front and rear elements.

Input Format

The first input line consists of an integer N, representing the number of customers to be inserted into the queue.

The second line consists of N space-separated integers, representing the customer IDs.

Output Format

The output prints "Front: X, Rear: Y" where X is the front element and Y is the rear element, after performing the dequeue operation.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 5
112 104 107 116 109

Output: Front: 104, Rear: 109

Answer

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


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

typedef struct Queue {
    Node* front;
    Node* rear;
} Queue;

void initQueue(Queue* q) {
    q->front = q->rear = NULL;
}
void enqueue(Queue* q, int value) {
    Node* temp = (Node*)malloc(sizeof(Node));
    temp->data = value;
    temp->next = NULL;
```

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

void dequeue(Queue* q) {
    if (q->front == NULL) return;

    Node* temp = q->front;
    q->front = q->front->next;

    if (q->front == NULL)
        q->rear = NULL;

    free(temp);
}

int getFront(Queue* q) {
    return (q->front != NULL) ? q->front->data : -1;
}

int getRear(Queue* q) {
    return (q->rear != NULL) ? q->rear->data : -1;
}

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

    Queue q;
    initQueue(&q);

    int id;
    for (int i = 0; i < N; i++) {
        scanf("%d", &id);
        enqueue(&q, id);
    }
```

```
    dequeue(&q);

    printf("Front: %d, Rear: %d\n", getFront(&q), getRear(&q));

    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

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

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

void enqueue(struct Node **front, struct Node **rear, int value) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (*rear == NULL) {
        *front = *rear=newNode;
        *rear = newNode;
        return;
    }

    (*rear)->next = newNode;
    *rear = newNode;
}

void findKthFromEnd(struct Node *front, int k) {
    struct Node *mainPtr = front;
    struct Node *refPtr = front;
    int count = 0;
    while (count < k) {
        if (refPtr == NULL) {
            printf("Queue does not have enough elements.\n");
            return;
        }
        refPtr = refPtr->next;
        count++;
```

```
    }

    while (refPtr != NULL) {
        mainPtr = mainPtr->next;
        refPtr = refPtr->next;
    }

    printf("%d\n", mainPtr->data);
}

int main() {
    int N, K;

    scanf("%d", &N);
    struct Node*front=NULL;
    struct Node*rear=NULL;

    for (int i = 0; i < N; i++) {
       int value;
        scanf("%d", &value);
        enqueue(&front, &rear, value);
    }

    scanf("%d", &K);


    findKthFromEnd(front, K);

    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

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


typedef struct Node {
    int ticket;
    struct Node* next;
} Node;


typedef struct Queue {
    Node* front;
    Node* rear;
} Queue;
```

```
void initQueue(Queue* q) {
    q->front = q->rear = NULL;
}


void enqueue(Queue* q, int ticket) {
    Node* temp = (Node*)malloc(sizeof(Node));
    temp->ticket = ticket;
    temp->next = NULL;

    if (q->rear == NULL) {
        q->front = q->rear = temp;
    } else {
        q->rear->next = temp;
        q->rear = temp;
    }
}


int calculateSum(Queue* q) {
    int sum = 0;
    Node* current = q->front;
    while (current != NULL) {
        sum += current->ticket;
        current = current->next;
    }
    return sum;
}


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

    Queue q;
    initQueue(&q);

    int ticket;
    for (int i = 0; i < N; i++) {
        scanf("%d", &ticket);
```

```
        enqueue(&q, ticket);
    }

    int total = calculateSum(&q);
    printf("%d\n", total);

    return 0;
}
```

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