Rajalakshmi Engineering College

Name: Lakshmi Narayanan S
Email: 241801133@rajalakshmi.edu.in
Roll no: 241801133
Phone: 9345832054
Branch: REC
Department: I AI & DS - AE
Batch: 2028
Degree: B.E - AI & DS

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

Sara builds a linked list-based queue and wants to dequeue and display all positive even numbers in the queue. The numbers are added at the end of the queue.

Help her by writing a program for the same.

Input Format

The first line of input consists of an integer N, representing the number of elements Sara wants to add to the queue.

The second line consists of N space-separated integers, each representing an element to be enqueued.

Output Format

The output prints space-separated the positive even integers from the queue, maintaining the order in which they were enqueued.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5

Output: 2 4

Answer

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

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

// Define a queue structure
struct Queue {
    struct Node* front;
    struct Node* rear;
};

// Function to initialize the queue
void initializeQueue(struct Queue* q) {
    q->front = q->rear = NULL;
}

// Function to check if the queue is empty
int isEmpty(struct Queue* q) {
    return (q->front == NULL);
}

// Function to enqueue an element into the queue
void enqueue(struct Queue* q, int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (isEmpty(q)) {
        q->front = q->rear = newNode;
    } else {
        q->rear->next = newNode;
        q->rear = newNode;
    }
}

// Function to dequeue an element and print positive even numbers
void dequeueAndDisplayEven(struct Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty\n");
        return;
    }

    struct Node* temp = q->front;
    while (temp != NULL) {
        if (temp->data > 0 && temp->data % 2 == 0) {
            printf("%d ", temp->data);
        }
        temp = temp->next;
    }
    printf("\n");
}

// Main function to drive the program
int main() {
    struct Queue q;
    int N, element;

    // Initialize the queue
    initializeQueue(&q);

    // Input number of elements in the queue
    scanf("%d", &N);

    // Input elements to enqueue
    for (int i = 0; i < N; i++) {
        scanf("%d", &element);
        enqueue(&q, element);
    }

    // Call the function to dequeue and display positive even numbers
    dequeueAndDisplayEven(&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

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

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

// Define the queue structure
struct Queue {
   struct Node* front;
   struct Node* rear;
};

// Function to initialize the queue
void initializeQueue(struct Queue* q) {
   q->front = q->rear = NULL;
}

// Function to enqueue an element into the queue
void enqueue(struct Queue* q, int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = NULL;

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

```
}

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

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

    // Move both pointers until the first pointer reaches the end
    while (first != NULL) {
        first = first->next;
        second = second->next;
    }

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

// Main function to drive the program
int main() {
    struct Queue q;
    int N, K, element;

    // Initialize the queue
    initializeQueue(&q);

    // Input number of elements in the queue
    scanf("%d", &N);

    // Input elements to enqueue
    for (int i = 0; i < N; i++) {
        scanf("%d", &element);
        enqueue(&q, element);
    }

    // Input K value
    scanf("%d", &K);
    // Find the Kth element from the end and print it
    int result = findKthFromEnd(&q, K);
    printf("%d\n", result);

    return 0;
}
```

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

3. 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

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

// Define the node structure for the linked list
struct Node {
    int data;
    struct Node* next;
};

// Function to initialize the queue
void initializeQueue(struct Node** front, struct Node** rear) {
    *front = *rear = NULL;
}

// Function to enqueue an element into the queue
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;
    } else {
        (*rear)->next = newNode;
        *rear = newNode;
    }
}

// Function to check if a value exists in the queue
int existsInQueue(struct Node* front, int value) {
    struct Node* temp = front;
    while (temp != NULL) {
        if (temp->data == value) {
            return 1;  // Value exists in queue
        }
        temp = temp->next;
    }
    return 0;  // Value does not exist
}

// Function to print the elements of the queue
void printQueue(struct Node* front) {
    struct Node* temp = front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Main function
int main() {
    struct Node* front = NULL;
    struct Node* rear = NULL;
    int N, request;

    // Input the number of requests
    scanf("%d", &N);

    // Initialize the queue
    initializeQueue(&front, &rear);

    // Input the requests and enqueue them
    for (int i = 0; i < N; i++) {
        scanf("%d", &request);
        if (!existsInQueue(front, request)) { // Only enqueue if the request is unique
            enqueue(&front, &rear, request);
        }
    }

    // Output the remaining unique requests in the queue
    printQueue(front);

    return 0;
}
```

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