Rajalakshmi Engineering College

Name: shobbika T
Email: 240701502@rajalakshmi.edu.in
Roll no: 240701502
Phone: 7305423247
Branch: REC
Department: I CSE FE
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

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

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Queue {
    struct Node* front;
    struct Node* rear;
};
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void initQueue(struct Queue* q) {
    q->front = q->rear = NULL;
}
void enqueue(struct Queue* q, int data) {
    struct Node* newNode = createNode(data);
    if (q->rear == NULL) {
        q->front = q->rear = newNode;
    } else {
        q->rear->next = newNode;
        q->rear = newNode;
    }
}
```

```
int dequeue(struct Queue* q) {
    if (q->front == NULL) {
        return -1;
    }
    struct Node* temp = q->front;
    int data = temp->data;
    q->front = q->front->next;
    if (q->front == NULL) {
        q->rear = NULL;
    }
    free(temp);
    return data;
}
int isEmpty(struct Queue* q) {
    return q->front == NULL;
}

int main() {
    int N, num;
    struct Queue q;
    initQueue(&q);
    scanf("%d", &N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &num);
        enqueue(&q, num);
    }
    while (!isEmpty(&q)) {
        int val = dequeue(&q);
        if (val > 0 && val % 2 == 0) {
            printf("%d ", val);
        }
    }

    return 0;
}
```

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

2. Problem Statement

Imagine you are developing a basic task management system for a small

team of software developers. Each task is represented by an integer, where positive integers indicate valid tasks and negative integers indicate erroneous tasks that need to be removed from the queue before processing.

Write a program using the queue with a linked list that allows the team to add tasks to the queue, remove all erroneous tasks (negative integers), and then display the valid tasks that remain in the queue.

Input Format

The first line consists of an integer N, representing the number of tasks to be added to the queue.

The second line consists of N space-separated integers, representing the tasks. Tasks can be both positive (valid) and negative (erroneous).

Output Format

The output displays the following format:

For each task enqueued, print a message "Enqueued: " followed by the task value.

The last line displays the "Queue Elements after Dequeue: " followed by removing all erroneous (negative) tasks and printing the valid tasks remaining in the queue in the order they were enqueued.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
12 -54 68 -79 53

Output: Enqueued: 12
Enqueued: -54
Enqueued: 68
Enqueued: -79
Enqueued: 53
Queue Elements after Dequeue: 12 68 53

Answer

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Queue {
    struct Node* front;
    struct Node* rear;
};
void initQueue(struct Queue* q) {
    q->front = q->rear = NULL;
}
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void enqueue(struct Queue* q, int data) {
    struct Node* newNode = createNode(data);
    if (q->rear == NULL) {
        q->front = q->rear = newNode;
    } else {
        q->rear->next = newNode;
        q->rear = newNode;
    }
    printf("Enqueued: %d\n", data);
}
int dequeue(struct Queue* q) {
    if (q->front == NULL)
        return -1;

    struct Node* temp = q->front;
    int value = temp->data;
    q->front = q->front->next;

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

    free(temp);
```

```
    return value;
}
void removeErroneousTasks(struct Queue* q) {
    struct Node* temp = q->front;
    struct Node* prev = NULL;

    while (temp != NULL) {
        if (temp->data < 0) {
            if (temp == q->front) {
                q->front = temp->next;
                if (q->rear == temp)
                    q->rear = NULL;
                free(temp);
                temp = q->front;
            } else {
                prev->next = temp->next;
                if (temp == q->rear)
                    q->rear = prev;
                free(temp);
                temp = prev->next;
            }
        } else {
            prev = temp;
            temp = temp->next;
        }
    }
}
void displayQueue(struct Queue* q) {
    struct Node* current = q->front;
    printf("Queue Elements after Dequeue: ");
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

int main() {
    int N, task;
    struct Queue q;
    initQueue(&q);
```

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

    removeErroneousTasks(&q);
    displayQueue(&q);

    return 0;
}
```

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

3. 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>

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

// Queue structure
struct Queue {
    struct Node* front;
    struct Node* rear;
};

// Initialize the queue
void initQueue(struct Queue* q) {
    q->front = q->rear = NULL;
}

// Create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Enqueue function
void enqueue(struct Queue* q, int data) {
    struct Node* newNode = createNode(data);
```

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

// Find the Kth element from the end using two-pointer approach
int findKthFromEnd(struct Queue* q, int k) {
    struct Node* first = q->front;
    struct Node* second = q->front;

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

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

    return second->data;
}

// Main function
int main() {
    struct Queue q;
    initQueue(&q);
    int N, K, value;

    // Read number of elements
    scanf("%d", &N);

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

```
    // Read K
    scanf("%d", &K);

    // Print the Kth element from the end
    int result = findKthFromEnd(&q, K);
    printf("%d\n", result);

    return 0;
}
```

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