Rajalakshmi Engineering College

Name: Pooja S
Email: 240701386@rajalakshmi.edu.in
Roll no: 240701386
Phone: 8838480229
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

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, *rear;
};
```

```
// Function to create a new node
struct Node* newNode(int data) {
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = data;
    temp->next = NULL;
    return temp;
}

// Function to create a new queue
struct Queue* createQueue() {
    struct Queue* q = (struct Queue*)malloc(sizeof(struct Queue));
    q->front = q->rear = NULL;
    return q;
}

// Enqueue function
void enqueue(struct Queue* q, int data) {
    struct Node* temp = newNode(data);
    if (q->rear == NULL) {
        q->front = q->rear = temp;
    } else {
        q->rear->next = temp;
        q->rear = temp;
    }
    printf("Enqueued: %d\n", data);
}

// Function to remove erroneous tasks (negative values)
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;
                free(temp);
                temp = q->front;
                if (q->front == NULL) {
                    q->rear = NULL;
                }
```

```
            } else {
                prev->next = temp->next;
                if (temp == q->rear) {
                    q->rear = prev;
                }
                free(temp);
                temp = prev->next;
            }
        } else {
            prev = temp;
            temp = temp->next;
        }
    }
}

// Display valid tasks
void displayQueue(struct Queue* q) {
    printf("Queue Elements after Dequeue: ");
    struct Node* temp = q->front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Main function
int main() {
    int n, value;
    scanf("%d", &n);
    struct Queue* q = createQueue();

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

    removeErroneousTasks(q);
    displayQueue(q);

    return 0;
}
```

Status : Correct *Marks : 10/10*

2. Problem Statement

Pathirana is a medical lab specialist who is responsible for managing blood count data for a group of patients. The lab uses a queue-based system to track the blood cell count of each patient. The queue structure helps in processing the data in a first-in-first-out (FIFO) manner.

However, Pathirana needs to remove the blood cell count that is positive even numbers from the queue using array implementation of queue, as they are not relevant to the specific analysis he is performing. The remaining data will then be used for further medical evaluations and reporting.

Input Format

The first line consists of an integer n, representing the number of a patient's blood cell count.

The second line consists of n space-separated integers, representing a blood cell count value.

Output Format

The output displays space-separated integers, representing the remaining blood cell count after removing the positive even numbers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
Output: 1 3 5

Answer

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

```
int main() {
    int n, i;
    int bloodCounts[15];     // Array to store input values
    int result[15];          // Array to store filtered output
    int resultIndex = 0;

    // Input the number of elements
    scanf("%d", &n);

    // Input the blood cell counts
    for (i = 0; i < n; i++) {
        scanf("%d", &bloodCounts[i]);
    }

    // Filter: Remove positive even numbers
    for (i = 0; i < n; i++) {
        if (!(bloodCounts[i] > 0 && bloodCounts[i] % 2 == 0)) {
            result[resultIndex++] = bloodCounts[i];
        }
    }

    // Print the remaining blood cell counts
    for (i = 0; i < resultIndex; i++) {
        printf("%d ", result[i]);
    }

    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

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

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
```

```
void enqueue(struct Node** front, struct Node** rear, int data) {
    struct Node* newNode = createNode(data);
    if (*rear == NULL) {
        *front = *rear = newNode;
        return;
    }
    (*rear)->next = newNode;
    *rear = newNode;
}

void removeDuplicates(struct Node* front) {
    struct Node* current = front;
    while (current != NULL) {
        struct Node* runner = current;
        while (runner->next != NULL) {
            if (runner->next->data == current->data) {
                struct Node* temp = runner->next;
                runner->next = runner->next->next;
                free(temp);
            } else {
                runner = runner->next;
            }
        }
        current = current->next;
    }
}

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

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

```
    for (int i = 0; i < N; i++) {
        int val;
        scanf("%d", &val);
        enqueue(&front, &rear, val);
    }
    removeDuplicates(front);
    display(front);
    return 0;
}
```

Status : Correct *Marks : 10/10*