Rajalakshmi Engineering College

Name: Navaneetha Krishnan
Email: 241901067@rajalakshmi.edu.in
Roll no: 241901067
Phone: 8939010233
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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>
 struct Node{
    int data;
    struct Node*next,*prev;
 }*front=NULL,*rear=NULL;
 void enqueue(int data){
    struct Node*newnode=(struct Node*)malloc(sizeof(struct Node));
    newnode->data=data;
    newnode->next=NULL;
    if(front==NULL)
      front=rear=newnode;
     else{
        rear->next=newnode;
        rear=newnode;
     }
 }
 void dequeue(){
    if(front!=NULL){
      struct Node*temp=front;
      while(temp!=NULL){
         if(temp->data%2!=0){
            temp=temp->next;
            continue;
         }
         else if(temp->data%2==0 && temp->data>0)
         printf("%d ",temp->data);
```

```
            temp=temp->next;
        }
    }
}
int main(){
    int n,data;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d ",&data);
        enqueue(data);
    }
    dequeue();
}
```

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

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

struct Node{
   int data;
   struct Node*next,*prev;
}*front=NULL,*rear=NULL;

void enqueue(int data){
   struct Node*newnode=(struct Node*)malloc(sizeof(struct Node));
   newnode->data=data;
   newnode->next=NULL;
   if(front==NULL)
   front=rear=newnode;
   else{
      rear->next=newnode;
      rear=newnode;
   }
}
void dequeue(){
    if(front!=NULL){
        struct Node*temp=front;
        while(temp!=NULL){
            if(temp->data >0){
                printf("%d ",temp->data);
                temp=temp->next;
            }
            else{
                temp=temp->next;
                continue;
            }
        }
    }
}
int main(){
    int n ,data;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&data);
        printf("Enqueued: %d\n",data);
        enqueue(data);
    }
    printf("Queue Elements after Dequeue: ");
    dequeue();
}
```

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>

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

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

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

void enqueue(Queue* q, int request) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   if (newNode == NULL) {
      perror("Memory allocation failed");
      exit(EXIT_FAILURE);
```

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

int exists(Queue* q, int request) {
    Node* current = q->front;
    while (current != NULL) {
        if (current->data == request) {
            return 1;
            }
        current = current->next;
    }
    return 0;
}

void removeDuplicatesAndPrint(Queue* q) {
    Queue uniqueQueue;
    initializeQueue(&uniqueQueue);
    Node* current = q->front;

    while (current != NULL) {
        if (!exists(&uniqueQueue, current->data)) {
            enqueue(&uniqueQueue, current->data);
        }
        current = current->next;
    }

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

    Node* currentUnique = uniqueQueue.front;
    while (currentUnique != NULL) {
        Node* next = currentUnique->next;
        free(currentUnique);
        currentUnique = next;
    }
}

void freeQueue(Queue* q) {
    Node* current = q->front;
    while (current != NULL) {
        Node* next = current->next;
        free(current);
        current = next;
    }
    q->front = NULL;
    q->rear = NULL;
}

int main() {
    int n;

    if (scanf("%d", &n) != 1) {
        fprintf(stderr, "Error reading the number of requests.\n");
        return 1;
    }

    if (n < 1 || n > 15) {
        fprintf(stderr, "N is out of the valid range.\n");
        return 1;
    }

    Queue requestQueue;
    initializeQueue(&requestQueue);

    for (int i = 0; i < n; i++) {
        int request;
        if (scanf("%d", &request) != 1) {
            fprintf(stderr, "Error reading a request.\n");
            return 1;
        }
        enqueue(&requestQueue, request);
    }

    removeDuplicatesAndPrint(&requestQueue);

    freeQueue(&requestQueue);
    return 0;
}
```

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