Rajalakshmi Engineering College

Name: Shremathi K
Email: 240701504@rajalakshmi.edu.in
Roll no:
Phone: 8870649491
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

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

```
#include <stdio.h>

#define MAX_SIZE 100

int bloodCounts[MAX_SIZE];
int front = -1, rear = -1;

void enqueue(int value) {
   if (rear == MAX_SIZE - 1) {
      printf("Queue is full!\n");
      return;
   }
   if (front == -1) {
      front = 0;
   }
   rear++;
   bloodCounts[rear] = value;
}

void processQueue() {
   for (int i = front; i <= rear; i++) {
      if (!(bloodCounts[i] > 0 && bloodCounts[i] % 2 == 0)) {
            printf("%d", bloodCounts[i]);
            if (i < rear) {
                printf(" ");
            }
        }
    }
    printf("\n");
}

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

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

    processQueue();

    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>

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

Node* front = NULL;
Node* rear = NULL;
```

```
void enqueue(int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->task = value;
    newNode->next = NULL;

    if (rear == NULL) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
    printf("Enqueued: %d\n", value);
}

void dequeue() {
    Node* temp = front;
    front = front->next;
    if (front == NULL) {
        rear = NULL;
    }
    free(temp);
}

void processQueue() {
    Node* current = front;
    printf("Queue Elements after Dequeue: ");
    while (current != NULL) {
        if (current->task > 0) {
            printf("%d ", current->task);
        }
        current = current->next;
    }
    printf("\n");
}

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

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

    processQueue();

    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 ticketNumber;
   struct Node* next;
} Node;

Node* front = NULL;
Node* rear = NULL;

void enqueue(int ticket) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->ticketNumber = ticket;
   newNode->next = NULL;

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

int calculateSum() {
   Node* current = front;
   int sum = 0;

   while (current != NULL) {
      sum += current->ticketNumber;
      current = current->next;
   }

   return sum;
}

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

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

   int totalSum = calculateSum();
   printf("%d\n", totalSum);

   return 0;
}
```

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