Rajalakshmi Engineering College

Name: Soameshwaran D
Email: 240701520@rajalakshmi.edu.in
Roll no: 240701520
Phone: 7358671540
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 : 20

Section 1 : Coding

1. 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

Status : Wrong ***Marks : 0/10***

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>

#define MAX_SIZE 15  // As per constraint: 1 ≤ N ≤ 15

int main() {
    int n, i;
    int queue[MAX_SIZE];
    int filtered[MAX_SIZE];  // To store remaining counts
    int filteredIndex = 0;

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

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

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

    // Print remaining counts
    for (i = 0; i < filteredIndex; i++) {
```

```
        printf("%d ", filtered[i]);
    }

    return 0;
}
```

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

3. 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>
```

```c
#include <stdlib.h>
// Node structure for the queue
struct Node {
  int data;
  struct Node* next;
};
// Queue front and rear pointers
struct Node* front = NULL;
struct Node* rear = NULL;
// Enqueue function
void enqueue(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;
  }

}
// Dequeue and print only positive even numbers
void displayPositiveEven() {
  struct Node* temp = front;
  while (temp != NULL) {
     if (temp->data > 0 && temp->data % 2 == 0) {
        printf("%d ", temp->data);
     }
     temp = temp->next;
  }
  printf("\n");
}
// Main function
int main() {
  int N, val;
  // Read number of elements
  scanf("%d", &N);
  // Read and enqueue elements
  for (int i = 0; i < N; i++) {
```

```
        scanf("%d", &val);
        enqueue(val);
    }
    // Display positive even numbers
    displayPositiveEven();
    return 0;
}
```

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