Rajalakshmi Engineering College

Name: SHIVANISREE K B
Email: 240701501@rajalakshmi.edu.in
Roll no: 240701501
Phone: 7358464804
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

John is working on a project to manage and analyze the data from various sensors in a manufacturing plant. Each sensor provides a sequence of integer readings, and John needs to process this data to get some insights. He wants to implement a queue to handle these sensor readings efficiently. The requirements are as follows:

Enqueue Operations: Each sensor reading needs to be added to the circular queue.Average Calculation: Calculate and print the average of every pair of consecutive sensor readings.Sum Calculation: Compute the sum of all sensor readings.Even and Odd Count: Count and print the number of even and odd sensor readings.

Assist John in implementing the program.

Input Format

The first input line contains an integer n, which represents the number of sensor readings.

The second line contains n space-separated integers, each representing a sensor reading.

Output Format

The first line should print "Averages of pairs:" followed by the averages of every pair of consecutive sensor readings, separated by spaces.

The second line should print "Sum of all elements: " followed by the sum of all sensor readings.

The third line should print "Number of even elements: " followed by the count of even sensor readings.

The fourth line should print "Number of odd elements: " followed by the count of odd sensor readings.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
Output: Averages of pairs:
1.5 2.5 3.5 4.5 3.0
Sum of all elements: 15
Number of even elements: 2
Number of odd elements: 3

Answer

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

#define MAX_SIZE 10

int main() {
```

```
    int n, i;
    int queue[MAX_SIZE];
    int sum = 0;
    int even_count = 0, odd_count = 0;


    scanf("%d", &n);


    for (i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
    }


    printf("Averages of pairs:\n");
    for (i = 0; i < n; i++) {
        float avg = (queue[i] + queue[(i + 1) % n]) / 2.0;
        printf("%.1f ", avg);
    }
    printf("\n");


    for (i = 0; i < n; i++) {
        sum += queue[i];
        if (queue[i] % 2 == 0)
            even_count++;
        else
            odd_count++;
    }

    printf("Sum of all elements: %d\n", sum);
    printf("Number of even elements: %d\n", even_count);
    printf("Number of odd elements: %d\n", odd_count);

    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>

#define MAX_SIZE 15

int main() {
    int n, i;
```

```
    int queue[MAX_SIZE];
    int filtered[MAX_SIZE];
    int filteredIndex = 0;


    scanf("%d", &n);


    for (i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
    }


    for (i = 0; i < n; i++) {
        if (!(queue[i] > 0 && queue[i] % 2 == 0)) {
            filtered[filteredIndex++] = queue[i];
        }
    }


    for (i = 0; i < filteredIndex; i++) {
        printf("%d ", filtered[i]);
    }

    return 0;
}
```

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

3. 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;
```

```
};

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


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;
   }

   printf("Enqueued: %d\n", value);
}


void removeErroneousTasks() {
   struct Node* current = front;
   struct Node* prev = NULL;

   while (current != NULL) {
      if (current->data < 0) {

         if (current == front) {
            front = current->next;
            free(current);
            current = front;
            if (front == NULL) rear = NULL;
         } else {
            prev->next = current->next;
            if (current == rear) rear = prev;
            free(current);
            current = prev->next;
         }
```

```
        } else {

            prev = current;
            current = current->next;
        }
    }
}


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

int main() {
    int n, i, task;

    scanf("%d", &n);

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

    removeErroneousTasks();
    displayQueue();

    return 0;
}
```

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