Rajalakshmi Engineering College

Name: Sherin Katherina
Email: 240701495@rajalakshmi.edu.in
Roll no: 240701495
Phone: 9150930353
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>

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

```
    int readings[10];
    for (int i = 0; i < n; i++) {
        scanf("%d", &readings[i]);
    }

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

    // Calculate sum, even and odd counts
    int sum = 0, even = 0, odd = 0;
    for (int i = 0; i < n; i++) {
        sum += readings[i];
        if (readings[i] % 2 == 0)
            even++;
        else
            odd++;
    }

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

    return 0;
}
```

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

2. 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>

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

// Queue structure
struct Queue {
   struct Node *front, *rear;
};

// Function to create a new node
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
```

```
    newNode->next = NULL;
    return newNode;
}

// Enqueue operation
void enqueue(struct Queue* q, int data) {
    struct Node* temp = createNode(data);
    if (q->rear == NULL) {
        q->front = q->rear = temp;
        return;
    }
    q->rear->next = temp;
    q->rear = temp;
}

// Remove duplicates
void removeDuplicates(struct Queue* q) {
    int seen[101] = {0}; // Since 1 ≤ request ≤ 100
    struct Node *curr = q->front, *prev = NULL;

    while (curr != NULL) {
        if (seen[curr->data]) {
            // Duplicate found, remove node
            prev->next = curr->next;
            if (curr == q->rear) // Update rear if needed
                q->rear = prev;
            struct Node* temp = curr;
            curr = curr->next;
            free(temp);
        } else {
            seen[curr->data] = 1;
            prev = curr;
            curr = curr->next;
        }
    }
}

// Print queue
void printQueue(struct Queue* q) {
    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 = {NULL, NULL};

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

    removeDuplicates(&q);
    printQueue(&q);

    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>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Queue {
    struct Node* front;
    struct Node* rear;
};
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void enqueue(struct Queue* q, int data) {
    struct Node* temp = createNode(data);
    if (q->rear == NULL) {
        q->front = q->rear = temp;
        return;
    }
    q->rear->next = temp;
    q->rear = temp;
}
```

```
void dequeuePositiveEvens(struct Queue* q) {
    struct Node* current = q->front;
    while (current != NULL) {
        if (current->data > 0 && current->data % 2 == 0) {
            printf("%d ", current->data);
        }
        current = current->next;
    }
    printf("\n");
}

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

    struct Queue q = {NULL, NULL};

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

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