Rajalakshmi Engineering College

Name: Sherwin G M
Email: 240701496@rajalakshmi.edu.in
Roll no: 240701496
Phone: 7708605966
Branch: REC
Department: I CSE FD
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

Fathima has been tasked with developing a program to manage a queue of customers waiting in line at a service center. Help her write a program simulating a queue data structure using a linked list.

Here is a description of the scenario and the required operations:

Enqueue: Add a customer to the end of the queue.Dequeue: Remove and discard a customer from the front of the queue.Display waiting customers: Display the front and rear customer IDs in the queue.

Write a program that enqueues all the customers into the queue, performs a dequeue operation, and prints the front and rear elements.

Input Format

The first input line consists of an integer N, representing the number of customers to be inserted into the queue.

The second line consists of N space-separated integers, representing the customer IDs.

Output Format

The output prints "Front: X, Rear: Y" where X is the front element and Y is the rear element, after performing the dequeue operation.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 5
112 104 107 116 109

Output: Front: 104, Rear: 109

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;

Queue* createQueue() {
    Queue* q = (Queue*)malloc(sizeof(Queue));
    q->front = q->rear = NULL;
    return q;
}

void enqueue(Queue* q, int data) {
```

```
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;

    if (q->rear == NULL) {
        q->front = q->rear = newNode;
        return;
    }

    q->rear->next = newNode;
    q->rear = newNode;
}

void dequeue(Queue* q) {
    if (q->front == NULL) {
        return;
    }

    Node* temp = q->front;
    q->front = q->front->next;

    if (q->front == NULL) {
        q->rear = NULL;
    }

    free(temp);
}

void printFrontRear(Queue* q) {
    if (q->front == NULL) {
        return;
    }
    printf("Front: %d, Rear: %d\n", q->front->data, q->rear->data);
}

int main() {
    int N, value;
    Queue* q = createQueue();

    scanf("%d", &N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &value);
```

```
        enqueue(q, value);
    }

    dequeue(q);
    printFrontRear(q);

    return 0;
}
```

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

2. 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

void enqueue(int queue[], int *size, int value) {
    if (*size < MAX_SIZE) {
        queue[*size] = value;
        (*size)++;
    }
}

void calculateAverages(int queue[], int size) {
    printf("Averages of pairs:\n");
    for (int i = 0; i < size - 1; i++) {
        printf("%.1f ", (queue[i] + queue[i + 1]) / 2.0);
    }
    if (size > 1) {
        printf("%.1f\n", (queue[0] + queue[size - 1]) / 2.0);
    }
```

```
}

int calculateSum(int queue[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += queue[i];
    }
    return sum;
}

void countEvenOdd(int queue[], int size, int *even, int *odd) {
    *even = *odd = 0;
    for (int i = 0; i < size; i++) {
        if (queue[i] % 2 == 0) {
            (*even)++;
        } else {
            (*odd)++;
        }
    }
}

int main() {
    int n, queue[MAX_SIZE], size = 0;

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

    calculateAverages(queue, size);

    printf("Sum of all elements: %d\n", calculateSum(queue, size));

    int even, odd;
    countEvenOdd(queue, size, &even, &odd);
    printf("Number of even elements: %d\n", even);
    printf("Number of odd elements: %d\n", odd);

    return 0;
}
```

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

```
#include <stdio.h>
#include <stdlib.h>

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

```
void enqueue(Node** front, Node** rear, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;

    if (*rear == NULL) {
        *front = *rear = newNode;
        return;
    }

    (*rear)->next = newNode;
    *rear = newNode;
}

void removeDuplicates(Node* front) {
    Node* current = front;
    while (current != NULL) {
        Node* prev = current;
        Node* temp = current->next;

        while (temp != NULL) {
            if (temp->data == current->data) {
                prev->next = temp->next;
                free(temp);
                temp = prev->next;
            } else {
                prev = temp;
                temp = temp->next;
            }
        }
        current = current->next;
    }
}

void printQueue(Node* front) {
    while (front != NULL) {
        printf("%d ", front->data);
        front = front->next;
    }
    printf("\n");
}
```

```
int main() {
    int N, value;
    Node* front = NULL;
    Node* rear = NULL;

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

    removeDuplicates(front);
    printQueue(front);

    return 0;
}
```

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