Rajalakshmi Engineering College

Name: NITHISH RAJ L
Email: 240701366@rajalakshmi.edu.in
Roll no: 2116240701366
Phone: 8072719523
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

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>

struct Node {
   int ticketNumber;
   struct Node* next;
};

void enqueue(struct Node** front, struct Node** rear, int ticketNumber) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->ticketNumber = ticketNumber;
   newNode->next = NULL;
   if (*rear == NULL) {
      *front = *rear = newNode;
      return;
   }
   (*rear)->next = newNode;
   *rear = newNode;
}

int calculateTotal(struct Node* front) {
   int sum = 0;
   struct Node* temp = front;
```

```
    while (temp != NULL) {
        sum += temp->ticketNumber;
        temp = temp->next;
    }
    return sum;
}

void displayQueue(struct Node* front) {
    struct Node* temp = front;
    while (temp != NULL) {
        printf("%d ", temp->ticketNumber);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct Node *front = NULL, *rear = NULL;
    int N, ticketNumber;

    scanf("%d", &N);

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

    int total = calculateTotal(front);
    printf("%d\n", total);

    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

```
#include <stdio.h>

#define MAX_SIZE 15

typedef struct {
   int items[MAX_SIZE];
   int front, rear;
} Queue;

void initQueue(Queue* q) {
   q->front = 0;
```

```
    q->rear = -1;
}

int isEmpty(Queue* q) {
    return q->rear < q->front;
}

void enqueue(Queue* q, int value) {
    if (q->rear == MAX_SIZE - 1) {
        printf("Queue is full\n");
        return;
    }
    q->items[++(q->rear)] = value;
}

int dequeue(Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty\n");
        return -1;
    }
    return q->items[(q->front)++];
}

void removePositiveEvens(Queue* q) {
    int i = q->front;
    while (i <= q->rear) {
        if (q->items[i] > 0 && q->items[i] % 2 == 0) {
            for (int j = i; j < q->rear; j++) {
                q->items[j] = q->items[j + 1];
            }
            q->rear--;
        } else {
            i++;
        }
    }
}

void displayQueue(Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty\n");
        return;
    }
    for (int i = q->front; i <= q->rear; i++) {
        printf("%d ", q->items[i]);
    }
    printf("\n");
}

int main() {
    Queue q;
    initQueue(&q);

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

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

    removePositiveEvens(&q);
    displayQueue(&q);

    return 0;
}
```

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

3. 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

```
#include <stdio.h>

#define MAX_SIZE 10
```

```
int getSum(int arr[], int n) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }
    return sum;
}

void countEvenOdd(int arr[], int n, int *evenCount, int *oddCount) {
    *evenCount = 0;
    *oddCount = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] % 2 == 0)
            (*evenCount)++;
        else
            (*oddCount)++;
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int readings[MAX_SIZE];
    for (int i = 0; i < n; i++) {
        scanf("%d", &readings[i]);
    }
    printf("Averages of pairs:\n");
    for (int i = 0; i < n - 1; i++) {
        printf("%.1f ", (readings[i] + readings[i + 1]) / 2.0);
    }
    if (n > 1) {
        printf("%.1f ", (readings[n - 1] + readings[0]) / 2.0);
    }
    printf("\n");
    int totalSum = getSum(readings, n);
    printf("Sum of all elements: %d\n", totalSum);
    int evenCount, oddCount;
    countEvenOdd(readings, n, &evenCount, &oddCount);
    printf("Number of even elements: %d\n", evenCount);
    printf("Number of odd elements: %d\n", oddCount);

    return 0;
}
```

Status : Correct *Marks : 10/10*