Rajalakshmi Engineering College

Name: Sowmyalakshmi   N
Email: 240701524@rajalakshmi.edu.in
Roll no: 240701524
Phone: 9003767185
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  // As per constraints: 1 ≤ n ≤ 10

int main() {
    int n, i;
    int queue[MAX_SIZE]; // Circular queue, but for n ≤ 10 a simple array will
suffice
    int sum = 0;
    int even_count = 0, odd_count = 0;

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

    // Input readings and enqueue operation
    for (i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
    }

    // Calculate averages of consecutive pairs (including last & first for circularity)
    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");

    // Sum, even count, and odd count
    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

Manoj is learning data structures and practising queues using linked lists. His professor gave him a problem to solve. Manoj started solving the program but could not finish it. So, he is seeking your assistance in solving it.

The problem is as follows: Implement a queue with a function to find the Kth element from the end of the queue.

Help Manoj with the program.

Input Format

The first line of input consists of an integer N, representing the number of elements in the queue.

The second line consists of N space-separated integers, representing the queue elements.

The third line consists of an integer K.

Output Format

The output prints an integer representing the Kth element from the end of the queue.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 6 7 5
3
Output: 6

Answer

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

// Define a node of the queue
struct Node {
    int data;
    struct Node* next;
};

// Define front and rear pointers
struct Node* front = NULL;
struct Node* rear = NULL;

// Function to enqueue an element
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;
    }
}

// Function to find Kth element from end
int findKthFromEnd(int k) {
    int length = 0;
    struct Node* temp = front;

    // Count the number of nodes
    while (temp != NULL) {
        length++;
        temp = temp->next;
    }

    // Position from start is (length - k)
    int posFromStart = length - k + 1;  // FIXED: Added +1

    temp = front;
    for (int i = 1; i < posFromStart; i++) {
        temp = temp->next;
    }

    return temp->data;
}
```

```
// Main function
int main() {
   int N, K, value;

   // Input number of elements
   scanf("%d", &N);

   // Input queue elements
   for (int i = 0; i < N; i++) {
      scanf("%d", &value);
      enqueue(value);
   }

   // Input K
   scanf("%d", &K);

   // Output the Kth element from the end
   int result = findKthFromEnd(K);
   printf("%d\n", result);

   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>

// 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***