Rajalakshmi Engineering College

Name: NEIL DANIEL A
Email: 240701356@rajalakshmi.edu.in
Roll no: 240701356
Phone: 8925059757
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

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>

void calculateAverages(int readings[], int n) {
    printf("Averages of pairs:\n");
    for (int i = 0; i < n; i++) {
        int nextIndex = (i + 1) % n;
```

```
        float average = (readings[i] + readings[nextIndex]) / 2.0;
        printf("%.1f ", average);
    }
    printf("\n");
}

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

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

int main() {
    int n;
    scanf("%d", &n);
    int readings[n];

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

    calculateAverages(readings, n);

    int sum = calculateSum(readings, n);
    printf("Sum of all elements: %d\n", sum);
    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***

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 100
int q[max],f=-1,r=-1;
void eq(int d){
   if(f==-1)
   f=0;
   r++;
   q[r]=d;
}
void bo(){
   for(int i=f;i<=r;i++){
      if(q[i]<0)
      printf("%d ",q[i]);
      else if(q[i]%2!=0)
      printf("%d ",q[i]);

   }
}
int main(){
   int n,val;
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      scanf("%d",&val);
      eq(val);
   }
   bo();
}
```

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

3. 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>

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;
}

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


void enqueue(struct Queue* queue, int data) {
    struct Node* newNode = createNode(data);
    if (queue->rear == NULL) {
        queue->front = queue->rear = newNode;
        return;
    }
    queue->rear->next = newNode;
    queue->rear = newNode;
}

int findKthFromEnd(struct Queue* queue, int K) {
    struct Node* main_ptr = queue->front;
    struct Node* ref_ptr = queue->front;

    for (int i = 0; i < K; i++) {
        if (ref_ptr == NULL) {
            return -1;
        }
        ref_ptr = ref_ptr->next;
    }

    while (ref_ptr != NULL) {
        main_ptr = main_ptr->next;
        ref_ptr = ref_ptr->next;
    }
```

```
    return main_ptr->data;
}

int main() {
    int N, K;

    scanf("%d", &N);

    struct Queue* queue = createQueue();

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

    scanf("%d", &K);
    int result = findKthFromEnd(queue, K);
    printf("%d\n", result);

    return 0;
}
```

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