Rajalakshmi Engineering College

Name: Nitin Aakash
Email: 240701370@rajalakshmi.edu.in
Roll no: 240701370
Phone: 9498349045
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

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

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

   (*rear)->next = newNode;
   *rear = newNode;
}
void findKthFromEnd(struct Node* front, int K) {
    struct Node* mainPtr = front;
    struct Node* refPtr = front;
    int count = 0;
    while (count < K) {
        if (refPtr == NULL) {
            printf("Queue does not have enough elements.\n");
            return;
        }
        refPtr = refPtr->next;
        count++;
    }
    while (refPtr != NULL) {
        mainPtr = mainPtr->next;
        refPtr = refPtr->next;
    }
    printf("%d\n", mainPtr->data);
}

int main() {
    int N, K;
    scanf("%d", &N);

    struct Node* front = NULL;
    struct Node* rear = NULL;
    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        enqueue(&front, &rear, value);
    }
    scanf("%d", &K);
    findKthFromEnd(front, K);

    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

```
#include <stdio.h>

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

   int sensor_readings[n];
   for (int i = 0; i < n; i++) {
      scanf("%d", &sensor_readings[i]);
   }
   int total_sum = 0;
   int even_count = 0;
   int odd_count = 0;
   printf("Averages of pairs:\n");
   for (int i = 0; i < n; i++) {
      total_sum += sensor_readings[i];
      if (sensor_readings[i] % 2 == 0) {
         even_count++;
      } else {
         odd_count++;
      }
      int next_index = (i + 1) % n;
      double average = (sensor_readings[i] + sensor_readings[next_index]) / 2.0;
      printf("%.1f ", average);
   }
   printf("\nSum of all elements: %d\n", total_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***

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>
struct Node {
   int data;
   struct Node* next;
};
void enqueue(struct Node** front, struct Node** rear, 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;
    }
}
void removeDuplicates(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        struct Node* runner = current;
        while (runner->next != NULL) {
            if (runner->next->data == current->data) {
                struct Node* temp = runner->next;
                runner->next = runner->next->next;
                free(temp);
            } else {
                runner = runner->next;
            }
        }
        current = current->next;
    }
}
void display(struct Node* front) {
    struct Node* temp = front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    int N;
    scanf("%d", &N);

    struct Node* front = NULL;
    struct Node* rear = NULL;

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

    return 0;
}
```

Status : Correct *Marks : 10/10*