Rajalakshmi Engineering College

Name: Subhalakshmi M
Email: 240701539@rajalakshmi.edu.in
Roll no: 240701539
Phone: 6379032776
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

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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

// Node structure
struct Node {
    int data;
    struct Node* next;
};

// Enqueue operation
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;
    }
}

// Remove duplicates from the queue
void removeDuplicates(struct Node* front) {
    struct Node* current = front;
    while (current != NULL) {
        struct Node* runner = current;
```

```
        while (runner->next != NULL) {
            if (runner->next->data == current->data) {
                struct Node* duplicate = runner->next;
                runner->next = runner->next->next;
                free(duplicate);
            } else {
                runner = runner->next;
            }
        }
        current = current->next;
    }
}

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

int main() {
    int n, value;
    struct Node* front = NULL;
    struct Node* rear = NULL;

    // Read the number of requests
    scanf("%d", &n);

    // Read the request values and enqueue them
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        enqueue(&front, &rear, value);
    }

    // Remove duplicates
    removeDuplicates(front);

    // Print the final queue
    printQueue(front);
```

```
    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>

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;
    }
}

int findKthFromEnd(struct Node* front, int k) {
    struct Node* temp = front;
    int length = 0;

    while (temp != NULL) {
        length++;
        temp = temp->next;
    }

    if (k > length) {
        return -1;
    }

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

```
    return temp->data;
}

int main() {
    int n, k, value;
    struct Node* front = NULL;
    struct Node* rear = NULL;

    scanf("%d",&n);

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

    scanf("%d", &k);

    int result = findKthFromEnd(front, 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

```
#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 printPositiveEven(struct Node* front) {
    while (front != NULL) {
        if (front->data > 0 && front->data % 2 == 0) {
            printf("%d ", front->data);
```

```
        }
        front = front->next;
    }
    printf("\n");
}

int main() {
    int n, value;
    struct Node* front = NULL;
    struct Node* rear = NULL;

    scanf("%d", &n);

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

    printPositiveEven(front);

    return 0;
}
```

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