Rajalakshmi Engineering College

Name: Pavithra J
Email: 240701381@rajalakshmi.edu.in
Roll no: 240701381
Phone: 9363364978
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 data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void enqueue(struct Node** head, struct Node** tail, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
    } else {
        (*tail)->next = newNode;
```

```
        *tail = newNode;
    }
}

int sumTickets(struct Node* head) {
    int sum = 0;
    struct Node* current = head;
    while (current != NULL) {
        sum += current->data;
        current = current->next;
    }
    return sum;
}

void freeList(struct Node* head) {
    struct Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

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

    struct Node* head = NULL;
    struct Node* tail = NULL;

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

    int total = sumTickets(head);
    printf("%d\n", total);

    freeList(head);

    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

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

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

    int queue[15];
    int front = 0, rear = 0;

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

    for (int i = front; i < rear; i++) {
        if (queue[i] > 0 && queue[i] % 2 == 0) {
            continue;
        } else {
            printf("%d ", queue[i]);
        }
    }

    return 0;
}
```

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

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

// Define the structure for a node in the linked list
struct Node {
    int data;
    struct Node* next;
};

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
```

```
}

// Function to enqueue a new node at the end of the linked list
void enqueue(struct Node** head, struct Node** tail, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
    } else {
        (*tail)->next = newNode;
        *tail = newNode;
    }
}

// Function to find the Kth element from the end of the linked list
int findKthFromEnd(struct Node* head, int k) {
    struct Node* current = head;
    int length = 0;

    // Calculate the length of the linked list
    while (current != NULL) {
        length++;
        current = current->next;
    }

    // Determine the position from the start
    int position = length - k;
    current = head;

    // Traverse to the position
    for (int i = 0; i < position; i++) {
        current = current->next;
    }

    return current->data;
}

// Function to free the memory allocated for the linked list
void freeList(struct Node* head) {
    struct Node* temp;
    while (head != NULL) {
        temp = head;
```

```
        head = head->next;
        free(temp);
    }
}

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

    struct Node* head = NULL;
    struct Node* tail = NULL;

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

    scanf("%d", &K);

    int result = findKthFromEnd(head, K);
    printf("%d\n", result);

    freeList(head);

    return 0;
}
```

Status : Correct *Marks : 10/10*