

Rajalakshmi Engineering College

Name: Ravi Sankar
Email: 240701424@rajalakshmi.edu.in
Roll no: 240701424
Phone: 8122932671
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;  
};
```

```
struct Queue {  
    struct Node* front;  
    struct Node* rear;  
};
```

```
void initQueue(struct Queue* q) {  
    q->front = q->rear = NULL;  
}
```

```
struct Node* createNode(int value) {  
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));  
    newNode->data = value;
```

```

    newNode->next = NULL;
    return newNode;
}

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

int findKthFromEnd(struct Queue* q, int k, int n) {
    int target = n - k;
    struct Node* temp = q->front;
    for (int i = 0; i < target; i++) {
        if (temp != NULL) {
            temp = temp->next;
        }
    }
    return temp != NULL ? temp->data : -1;
}

int main() {
    struct Queue q;
    initQueue(&q);

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

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

    int k;
    scanf("%d", &k);

    int result = findKthFromEnd(&q, k, n);

```

```
printf("%d\n", result);  
return 0;  
}
```

Status : Correct

Marks : 10/10

2. Problem Statement

Imagine you are developing a basic task management system for a small team of software developers. Each task is represented by an integer, where positive integers indicate valid tasks and negative integers indicate erroneous tasks that need to be removed from the queue before processing.

Write a program using the queue with a linked list that allows the team to add tasks to the queue, remove all erroneous tasks (negative integers), and then display the valid tasks that remain in the queue.

Input Format

The first line consists of an integer N, representing the number of tasks to be added to the queue.

The second line consists of N space-separated integers, representing the tasks. Tasks can be both positive (valid) and negative (erroneous).

Output Format

The output displays the following format:

For each task enqueued, print a message "Enqueued: " followed by the task value.

The last line displays the "Queue Elements after Dequeue: " followed by removing all erroneous (negative) tasks and printing the valid tasks remaining in the queue in the order they were enqueued.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

12 -54 68 -79 53

Output: Enqueued: 12

Enqueued: -54

Enqueued: 68

Enqueued: -79

Enqueued: 53

Queue Elements after Dequeue: 12 68 53

Answer

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

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

```
struct Node {  
    int data;  
    struct Node* next;  
};
```

```
struct Queue {  
    struct Node *front, *rear;  
};
```

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

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

```
void enqueue(struct Queue* q, int value) {  
    struct Node* temp = newNode(value);  
    if (q->rear == NULL) {  
        q->front = q->rear = temp;
```

```

    } else {
        q->rear->next = temp;
        q->rear = temp;
    }
    printf("Enqueued: %d\n", value);
}

```

```

void removeNegatives(struct Queue* q) {
    struct Node *current = q->front, *prev = NULL;

```

```

    while (current != NULL) {
        if (current->data < 0) {
            if (current == q->front) {
                q->front = current->next;
                if (q->front == NULL) {
                    q->rear = NULL;
                }
                free(current);
                current = q->front;
            } else {
                prev->next = current->next;
                if (current == q->rear) {
                    q->rear = prev;
                }
                free(current);
                current = prev->next;
            }
        } else {
            prev = current;
            current = current->next;
        }
    }
}

```

```

void displayQueue(struct Queue* q) {
    printf("Queue Elements after Dequeue:");
    struct Node* temp = q->front;
    if (!temp) {
        printf("\n");
        return;
    }
    while (temp) {

```

```

        printf(" %d", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

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

    struct Queue* q = createQueue();
    for (int i = 0; i < N; i++) {
        int task;
        scanf("%d", &task);
        enqueue(q, task);
    }

    removeNegatives(q);
    displayQueue(q);

    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;
};
```

```
struct Queue {
    struct Node* front;
    struct Node* rear;
};
```

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

```
void initQueue(struct Queue* q) {
    q->front = q->rear = NULL;
}
```

```
void enqueue(struct Queue* q, int value) {
    struct Node* temp = createNode(value);
```



```

    if (q->rear == NULL) {
        q->front = q->rear = temp;
    } else {
        q->rear->next = temp;
        q->rear = temp;
    }
}

void displayPositiveEven(struct Queue* q) {
    struct Node* temp = q->front;
    while (temp != NULL) {
        if (temp->data > 0 && temp->data % 2 == 0) {
            printf("%d ", temp->data);
        }
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int N, value;
    struct Queue q;
    initQueue(&q);

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

    displayPositiveEven(&q);
    return 0;
}

```

Status : Correct

Marks : 10/10