# Rajalakshmi Engineering College

Name: Pranav Narayanan
Email: 240701393@rajalakshmi.edu.in
Roll no:
Phone: 9500579427
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***

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

// Define the node structure
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
```

```
}

// Function to enqueue elements
void enqueue(Node** front, Node** rear, int data) {
    Node* newNode = createNode(data);
    if (*rear == NULL) {
        *front = *rear = newNode;
    } else {
        (*rear)->next = newNode;
        *rear = newNode;
    }
}

// Function to find the Kth element from the end
int findKthFromEnd(Node* front, int k, int n) {
    if(k > n || k <= 0) return -1;
    int posFromStart = n - k + 1;
    Node* temp = front;
    for (int i = 1; i < posFromStart; i++) {
        temp = temp->next;
    }
    return temp->data;
}

// Main function
int main() {
    int n, k, data;
    scanf("%d", &n);

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

    // Read queue elements
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        enqueue(&front, &rear, data);
    }

    // Read value of K
    scanf("%d", &k);

    // Output the Kth element from the end
```

```
    int result = findKthFromEnd(front, k, n);
    printf("%d\n", result);

    return 0;
}
```

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

2. 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***

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

```c
#include <stdlib.h>

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

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to enqueue elements
void enqueue(Node** front, Node** rear, int data) {
    Node* newNode = createNode(data);
    if (*rear == NULL) {
        *front = *rear = newNode;
    } else {
        (*rear)->next = newNode;
        *rear = newNode;
    }
}

// Function to dequeue and print positive even numbers
void displayPositiveEven(Node* front) {
    Node* current = front;
    while (current != NULL) {
        if (current->data > 0 && current->data % 2 == 0) {
            printf("%d ", current->data);
        }
        current = current->next;
    }
    printf("\n");
}
```

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

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

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

    displayPositiveEven(front);

    return 0;
}
```

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

3. 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***

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

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

// Create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed\n");
        exit(1);
    }
```

```
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Enqueue function
void enqueue(Node** front, Node** rear, int data) {
    Node* newNode = createNode(data);
    if (*rear == NULL) {
        *front = *rear = newNode;
    } else {
        (*rear)->next = newNode;
        *rear = newNode;
    }
    printf("Enqueued: %d ", data);
}

// Function to remove erroneous (negative) tasks
void removeErroneousTasks(Node** front) {
    // Remove negative nodes from the beginning
    while (*front && (*front)->data < 0) {
        Node* temp = *front;
        *front = (*front)->next;
        free(temp);
    }

    // Traverse and remove negative nodes in the middle
    Node* current = *front;
    while (current && current->next) {
        if (current->next->data < 0) {
            Node* temp = current->next;
            current->next = current->next->next;
            free(temp);
        } else {
            current = current->next;
        }
    }
}

// Function to display queue
void displayQueue(Node* front) {
    printf("Queue Elements after Dequeue: ");
```

```
    Node* temp = front;
    while (temp) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Main function
int main() {
    int n, task;
    scanf("%d", &n);

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

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

    printf("\n");

    removeErroneousTasks(&front);
    displayQueue(front);

    return 0;
}
```

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