Rajalakshmi Engineering College

Name: Jeffery Antony J
Email: 241901041@rajalakshmi.edu.in
Roll no: 241901041
Phone: 7305663808
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

You are tasked with developing a simple ticket management system for a customer support department. In this system, customers submit support tickets, which are processed in a First-In-First-Out (FIFO) order. The system needs to handle the following operations:

Ticket Submission (Enqueue Operation): New tickets are submitted by customers. Each ticket is assigned a unique identifier (represented by an integer). When a new ticket arrives, it should be added to the end of the queue.

Ticket Processing (Dequeue Operation): The support team processes tickets in the order they are received. The ticket at the front of the queue is processed first. After processing, the ticket is removed from the queue.

Display Ticket Queue: The system should be able to display the current state of the ticket queue, showing the sequence of ticket identifiers from front to rear.

Input Format

The first input line contains an integer n, the number of tickets submitted by customers.

The second line consists of a single integer, representing the unique identifier of each submitted ticket, separated by a space.

Output Format

The first line displays the "Queue: " followed by the ticket identifiers in the queue after all tickets have been submitted.

The second line displays the "Queue After Dequeue: " followed by the ticket identifiers in the queue after processing (removing) the ticket at the front.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 6
14 52 63 95 68 49
Output: Queue: 14 52 63 95 68 49
Queue After Dequeue: 52 63 95 68 49

Answer

```c
#include <stdio.h>

#define MAX 20

int main() {
    int n, queue[MAX];
    int front = 0, rear = -1;


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

```c
    for (int i = 0; i < n; i++) {
        int ticket_id;
        scanf("%d", &ticket_id);
        if (rear < MAX - 1) {
            queue[++rear] = ticket_id;
        }
    }

    printf("Queue: ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");

    // Dequeue the front ticket
    if (front <= rear) {
        front++;  // Just increment front to simulate dequeue
    }

    // Display the queue after dequeue
    printf("Queue After Dequeue: ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");

    return 0;
}
```

Status : Correct   *Marks : 10/10*

2. Problem Statement

Sharon is developing a queue using an array. She wants to provide the
functionality to find the Kth largest element. The queue should support the
addition and retrieval of the Kth largest element effectively. The maximum
capacity of the queue is 10.

Assist her in 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.

The third line consists of an integer K.

Output Format

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

The last line prints "The [K]th largest element: " followed by the Kth largest element.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 45 93 87 25
4
Output: Enqueued: 23
Enqueued: 45
Enqueued: 93
Enqueued: 87
Enqueued: 25
The 4th largest element: 25

Answer

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

#define MAX 10

// Function to compare two integers for descending sort
```

```c
int compare(const void *a, const void *b) {
    return (*(int *)b - *(int *)a);
}

int main() {
    int N, K;
    int queue[MAX];

    // Read number of elements
    scanf("%d", &N);

    // Check for valid N
    if (N > MAX) {
        printf("Queue overflow. Maximum size is 10.\n");
        return 1;
    }

    // Read queue elements and enqueue
    for (int i = 0; i < N; i++) {
        scanf("%d", &queue[i]);
        printf("Enqueued: %d\n", queue[i]);
    }

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

    // Sort the queue in descending order to find the Kth largest
    qsort(queue, N, sizeof(int), compare);

    // Print the Kth largest element
    if (K <= N) {
        printf("The %dth largest element: %d\n", K, queue[K - 1]);
    } else {
        printf("Not enough elements in the queue to find the %dth largest.\n", K);
    }

    return 0;
}
```

Status : Correct   *Marks : 10/10*

3. Problem Statement

You've been assigned the challenge of developing a queue data structure using a linked list.

The program should allow users to interact with the queue by enqueuing positive integers and subsequently dequeuing and displaying elements.

Input Format

The input consists of a series of integers, one per line. Enter positive integers into the queue.

Enter -1 to terminate input.

Output Format

The output prints the space-separated dequeued elements.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1
2
3
4
-1
Output: Dequeued elements: 1 2 3 4

Answer

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

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

```c
// Queue front and rear pointers
struct Node* front = NULL;
struct Node* rear = NULL;

// Enqueue operation
void enqueue(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) return;

    newNode->data = value;
    newNode->next = NULL;

    if (rear == NULL) { // Empty queue
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
}

// Dequeue operation and printing
void dequeueAndPrint() {
    printf("Dequeued elements: ");
    while (front != NULL) {
        printf("%d ", front->data);
        struct Node* temp = front;
        front = front->next;
        free(temp);
    }
    printf("\n");
}

int main() {
    int value;

    // Read input until -1
    while (1) {
        scanf("%d", &value);
        if (value == -1)
            break;
        if (value >= 0) // Enqueue only non-negative values
            enqueue(value);
```

```
    }

    // Output dequeued elements
    dequeueAndPrint();

    return 0;
}
```

Status : Correct   *Marks : 10/10*

4.   Problem Statement

Amar is working on a project where he needs to implement a special type
of queue that allows selective dequeuing based on a given multiple. He
wants to efficiently manage a queue of integers such that only elements
not divisible by a given multiple are retained in the queue after a selective
dequeue operation.

Implement a program to assist Amar in managing his selective queue.

Example

Input:

5

10 2 30 4 50

5

Output:

Original Queue: 10 2 30 4 50

Queue after selective dequeue: 2 4

Explanation:

After selective dequeue with a multiple of 5, the elements that are
multiples of 5 should be removed. Therefore, only 10, 30, and 50 should be
removed from the queue. The updated Queue is 2 4.

Input Format

The first line contains an integer n, representing the number of elements initially present in the queue.

The second line contains n space-separated integers, representing the elements of the queue.

The third line contains an integer multiple, representing the divisor for selective dequeue operation.

Output Format

The first line of output prints "Original Queue: " followed by the space-separated elements in the queue before the dequeue operation.

The second line prints "Queue after selective dequeue: " followed by the remaining space-separated elements in the queue, after deleting elements that are the multiples of the specified number.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
10 2 30 4 50
5
Output: Original Queue: 10 2 30 4 50
Queue after selective dequeue: 2 4

Answer

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

int main() {
    int n, multiple;

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

    int queue[50];      // Original queue
    int result[50];     // Queue after selective dequeue
    int count = 0;
```

```c
    // Read elements into the queue
    for (int i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
    }

    // Read the multiple for selective dequeue
    scanf("%d", &multiple);

    // Print original queue
    printf("Original Queue: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");

    // Filter out elements divisible by the multiple
    for (int i = 0; i < n; i++) {
        if (queue[i] % multiple != 0) {
            result[count++] = queue[i];
        }
    }

    // Print the updated queue
    printf("Queue after selective dequeue: ");
    for (int i = 0; i < count; i++) {
        printf("%d ", result[i]);
    }
    printf("\n");

    return 0;
}
```

Status : Correct   *Marks : 10/10*

5. Problem Statement

Guide Harish in developing a simple queue system for a customer service center. The customer service center can handle up to 25 customers at a time. The queue needs to support basic operations such as adding a customer to the queue, serving a customer (removing them from the

queue), and displaying the current queue of customers.

Use an array for implementation.

Input Format

The first line of the input consists of an integer N, the number of customers arriving at the service center.

The second line consists of N space-separated integers, representing the customer IDs in the order they arrive.

Output Format

After serving the first customer in the queue, display the remaining customers in the queue.

If a dequeue operation is attempted on an empty queue, display "Underflow".

If the queue is empty, display "Queue is empty".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
101 102 103 104 105
Output: 102 103 104 105

Answer

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

#define MAX 25

int main() {
    int queue[MAX];
    int N;

    // Read the number of customers
    scanf("%d", &N);
```

```c
    // Handle invalid input
    if (N > MAX) {
        printf("Queue overflow. Maximum capacity is 25.\n");
        return 1;
    }

    // Read customer IDs if N > 0
    if (N > 0) {
        for (int i = 0; i < N; i++) {
            scanf("%d", &queue[i]);
        }

        // Dequeue the first customer (simulate by shifting)
        for (int i = 1; i < N; i++) {
            printf("%d ", queue[i]);
        }
        printf("\n");
    } else {
        // N is 0, check for underflow
        int dummy;
        scanf("%d", &dummy);  // Read the placeholder zero
        printf("Underflow\n");
        printf("Queue is empty\n");
    }

    return 0;
}
```

Status : Correct   *Marks : 10/10*