Rajalakshmi Engineering College

Name: LIRESH NV
Email: 241501100@rajalakshmi.edu.in
Roll no: 241501100
Phone: 9840466142
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In an office setting, a print job management system is used to efficiently handle and process print jobs. The system is implemented using a queue data structure with an array.

The program provides the following operations:

Enqueue Print Job: Add a print job with a specified number of pages to the end of the queue.Dequeue Print Job: Remove and process the next print job in the queue.Display Queue: Display the print jobs in the queue

The program should ensure that print jobs are processed in the order they are received.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Enqueue the print job into the queue. If the choice is 1, the following input is a space-separated integer, representing the pages to be enqueued into the queue.

Choice 2: Dequeue a print job from the queue.

Choice 3: Display the print jobs in the queue.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the queue:

If the choice is 1:

1. Insert the given page into the queue and display "Print job with [page] pages is enqueued." where [page] is the number of pages that are inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue."

If the choice is 2:

1. Dequeue a page from the queue and display "Processing print job: [page] pages" where [page] is the corresponding page that is dequeued.
2. If the queue is empty without any elements, print "Queue is empty."

If the choice is 3:

1. The output prints "Print jobs in the queue: " followed by the space-separated pages present in the queue.
2. If there are no elements in the queue, print "Queue is empty."

If the choice is 4:

1. Exit the program and print "Exiting program"

If any other choice is entered, the output prints "Invalid option."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1
10
1
20
1
30
1
40
1
50
1
60
3
2
3
4

Output: Print job with 10 pages is enqueued.
Print job with 20 pages is enqueued.
Print job with 30 pages is enqueued.
Print job with 40 pages is enqueued.
Print job with 50 pages is enqueued.
Queue is full. Cannot enqueue.
Print jobs in the queue: 10 20 30 40 50
Processing print job: 10 pages
Print jobs in the queue: 20 30 40 50
Exiting program

Answer

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

#define MAX_SIZE 5  // Maximum size of the queue
```

```
// Queue structure
struct Queue {
    int queue[MAX_SIZE];
    int front;
    int rear;
};

// Initialize the queue
void initQueue(struct Queue* q) {
    q->front = -1;
    q->rear = -1;
}

// Check if the queue is full
int isFull(struct Queue* q) {
    return (q->rear == MAX_SIZE - 1);
}

// Check if the queue is empty
int isEmpty(struct Queue* q) {
    return (q->front == -1 || q->front > q->rear);
}

// Enqueue operation: Add a print job to the queue
void enqueue(struct Queue* q, int pages) {
    if (isFull(q)) {
        printf("Queue is full. Cannot enqueue.\n");
    } else {
        if (q->front == -1) {
            q->front = 0;  // Queue was empty
        }
        q->rear++;
        q->queue[q->rear] = pages;
        printf("Print job with %d pages is enqueued.\n", pages);
    }
}

// Dequeue operation: Remove and process the front print job
void dequeue(struct Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty.\n");
    } else {
```

```
        int pages = q->queue[q->front];
        printf("Processing print job: %d pages\n", pages);
        q->front++;
        if (q->front > q->rear) {
            q->front = q->rear = -1;  // Reset the queue if it becomes empty
        }
    }
}

// Display the print jobs in the queue
void displayQueue(struct Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty.\n");
    } else {
        printf("Print jobs in the queue: ");
        for (int i = q->front; i <= q->rear; i++) {
            printf("%d ", q->queue[i]);
        }
        printf("\n");
    }
}

// Main function to interact with the queue based on user input
int main() {
    struct Queue q;
    initQueue(&q);

    int choice, pages;

    while (1) {
        scanf("%d", &choice);

        switch (choice) {
            case 1: // Enqueue
                scanf("%d", &pages);
                enqueue(&q, pages);
                break;

            case 2: // Dequeue
                dequeue(&q);
                break;
```

```
            case 3: // Display queue
                displayQueue(&q);
                break;

            case 4: // Exit
                printf("Exiting program\n");
                return 0;

            default:
                printf("Invalid option.\n");
                break;
        }
    }

    return 0;
}
```

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