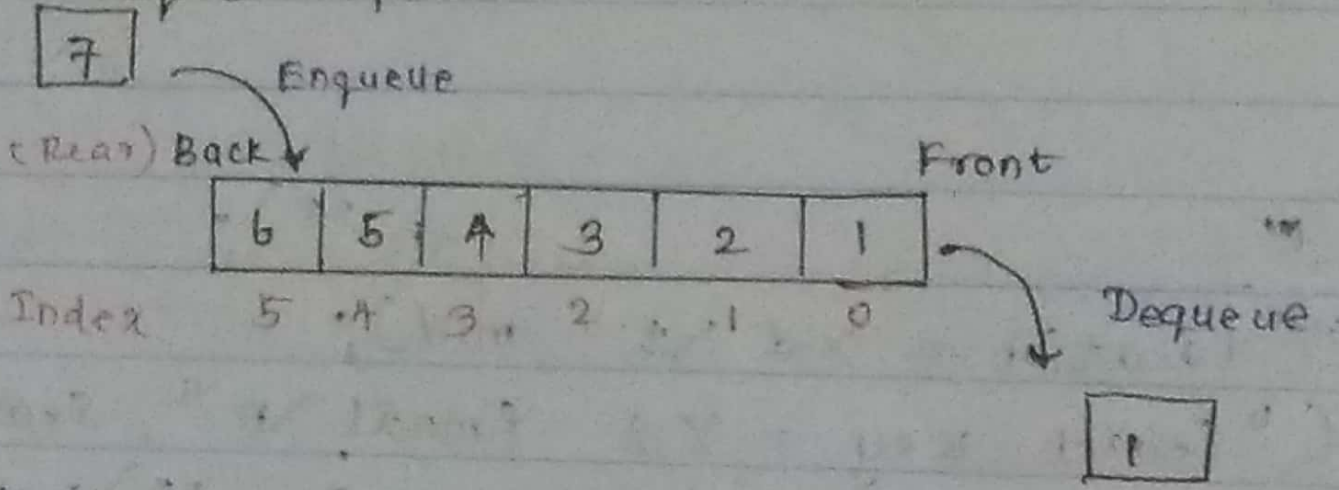
Tutorial 5

1 Draw the logical representation of queue?

For an example, if we consider the following linear number sequence, ~~the~~



According to the above representation, Queue follows LIFO method.

CamScanner

② There are 3 instances in which queue could be empty. Mention those.

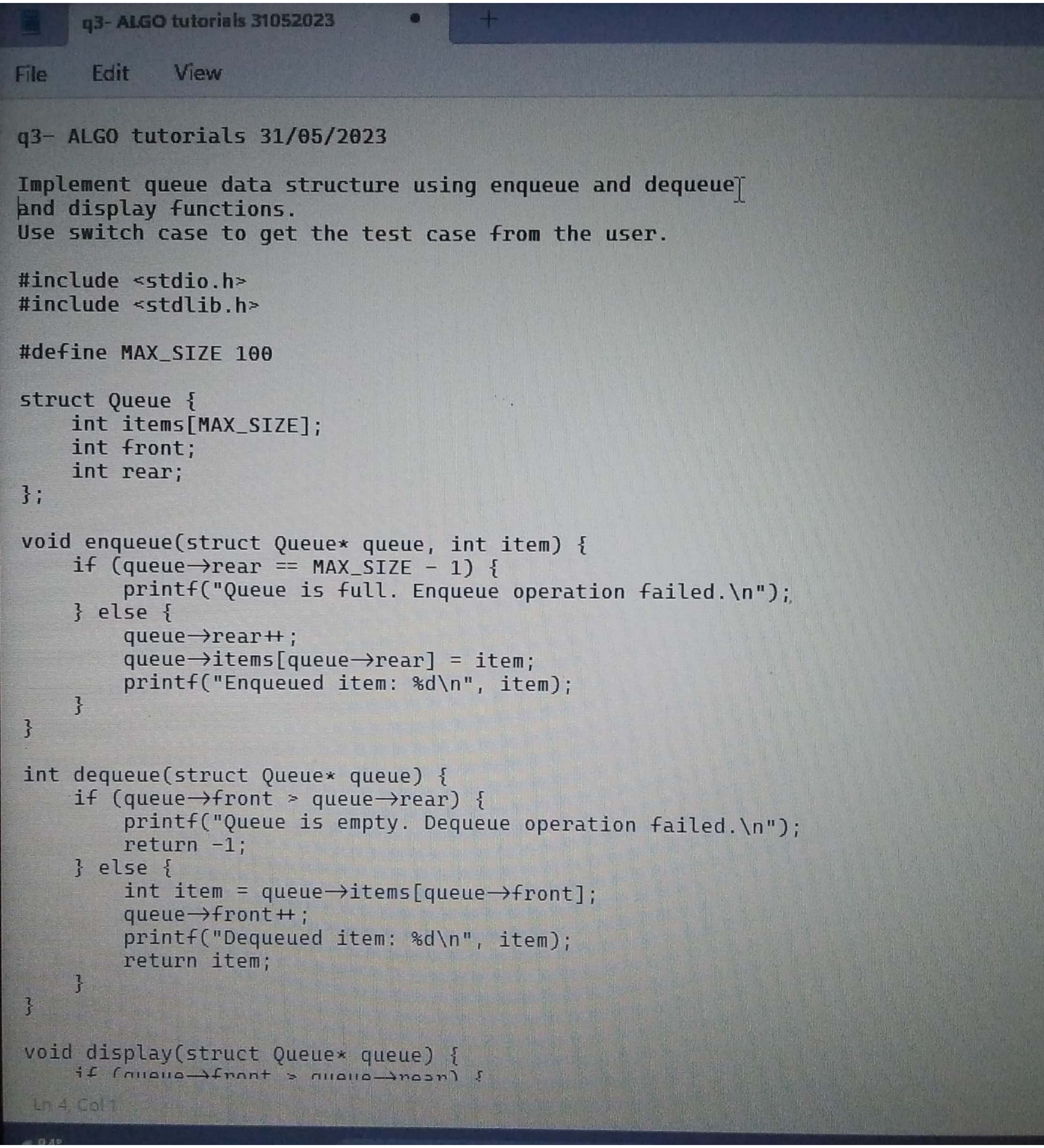
▷. The initial stage of a queue -

Answers

- * Front is at -1
- * front is greater than rear. - In circular queues.
- * Front is equal to rear. - In linear queues.

③ Implement queue data structure using enqueue() dequeue() and display() functions. Use switch case to get the test cases from the user.

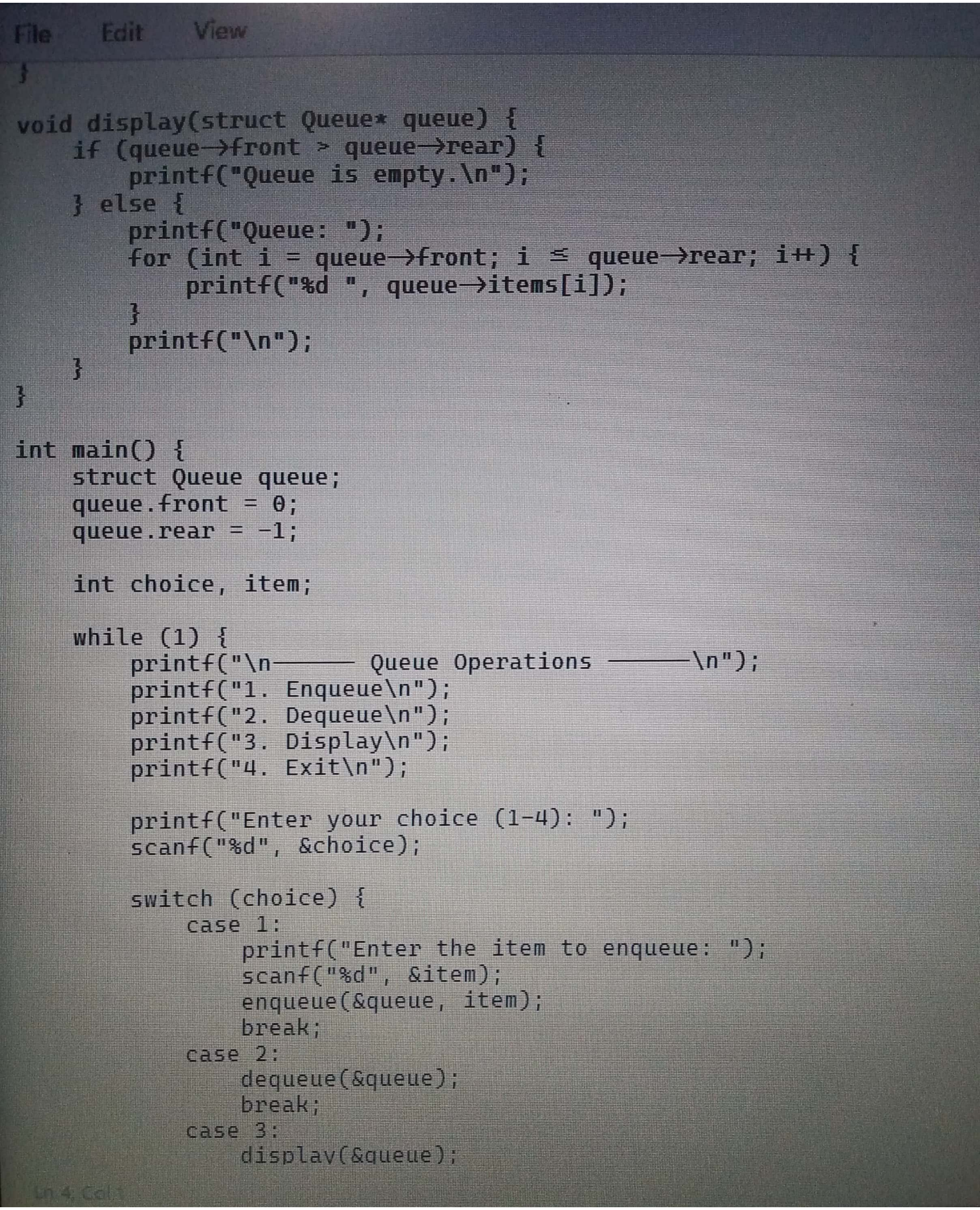
CS CamScanner



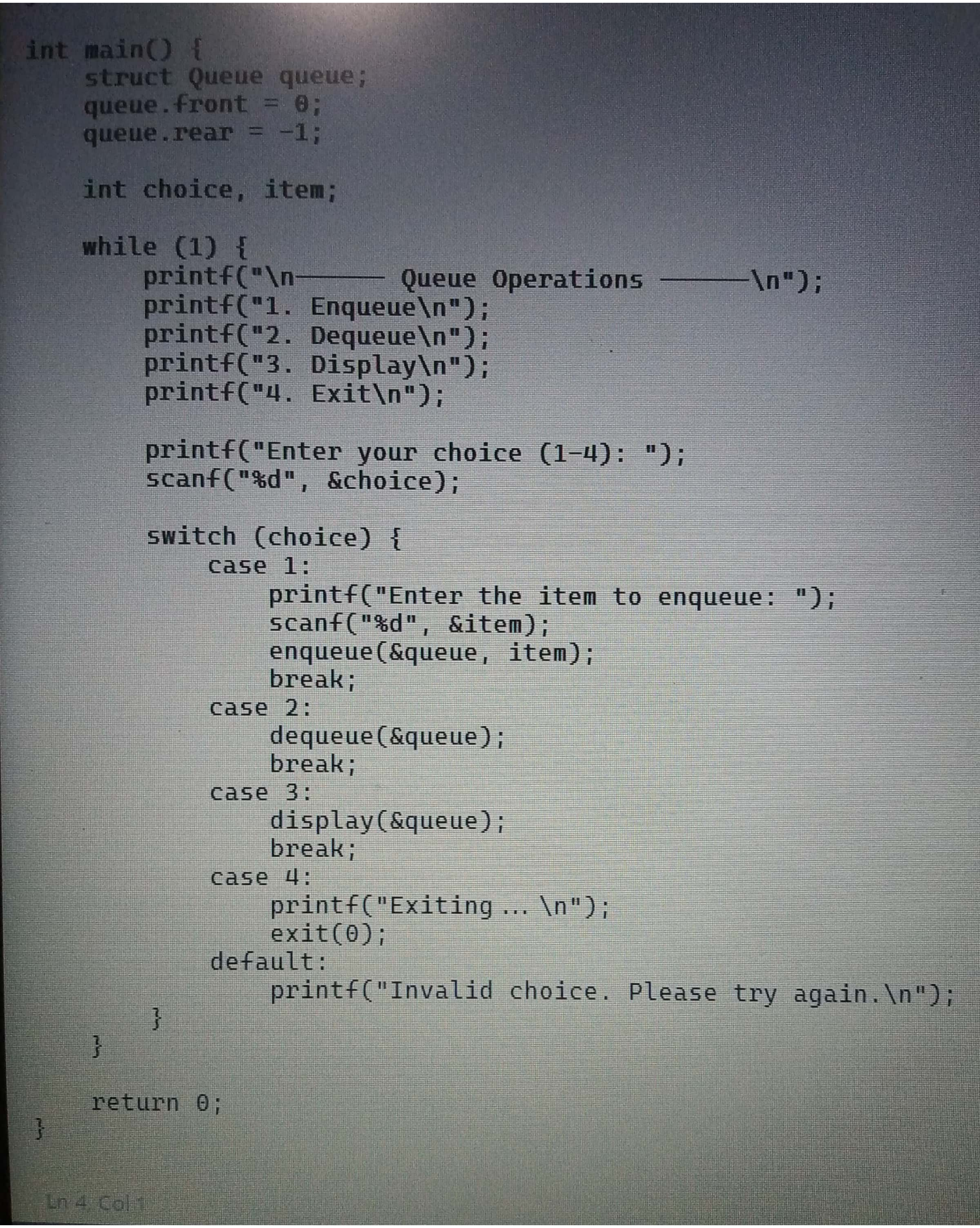
q3- ALGO tutorials 31/05/2023

Implement queue data structure using enqueue and dequeue
and display functions.
Use switch case to get the test case from the user.

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

#define MAX_SIZE 100

struct Queue {
    int items[MAX_SIZE];
    int front;
    int rear;
};

void enqueue(struct Queue* queue, int item) {
    if (queue->rear == MAX_SIZE - 1) {
        printf("Queue is full. Enqueue operation failed.\n");
    } else {
        queue->rear++;
        queue->items[queue->rear] = item;
        printf("Enqueued item: %d\n", item);
    }
}

int dequeue(struct Queue* queue) {
    if (queue->front > queue->rear) {
        printf("Queue is empty. Dequeue operation failed.\n");
        return -1;
    } else {
        int item = queue->items[queue->front];
        queue->front++;
        printf("Dequeued item: %d\n", item);
        return item;
    }
}

void display(struct Queue* queue) {
    if (queue->front > queue->rear) {
```



```
}

void display(struct Queue* queue) {
    if (queue->front > queue->rear) {
        printf("Queue is empty.\n");
    } else {
        printf("Queue: ");
        for (int i = queue->front; i <= queue->rear; i++) {
            printf("%d ", queue->items[i]);
        }
        printf("\n");
    }
}

int main() {
    struct Queue queue;
    queue.front = 0;
    queue.rear = -1;

    int choice, item;

    while (1) {
        printf("\n------ Queue Operations ------\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");

        printf("Enter your choice (1-4): ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the item to enqueue: ");
                scanf("%d", &item);
                enqueue(&queue, item);
                break;
            case 2:
                dequeue(&queue);
                break;
            case 3:
                display(&queue);
```



```
int main() {
    struct Queue queue;
    queue.front = 0;
    queue.rear = -1;

    int choice, item;

    while (1) {
        printf("\n——— Queue Operations ———\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");

        printf("Enter your choice (1-4): ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the item to enqueue: ");
                scanf("%d", &item);
                enqueue(&queue, item);
                break;
            case 2:
                dequeue(&queue);
                break;
            case 3:
                display(&queue);
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```