Part A

Ex : 4

Title: Design, develop, and implement an efficient use of memory data structure of Integers that follows the FIFO principle and demonstrates its operations.

Problem Description: The program creates the data structure and performs operations like inserting an element, deleting an element, and displaying the status of the data structure.

Method: In this program, create the data structure and support the program with appropriate functions for each of the above operations. Demonstrate Overflow and Underflow situations and support the program with appropriate functions for each of the above operations.

Theory Reference: Module 2

Explanation:

1. **Dynamic Memory Allocation:**
   - The size of the array (queue) is determined at runtime based on user input (max).
   - The array q is dynamically allocated using malloc().

2. **Queue Data Structure:**
   - A queue is a linear data structure that follows the First In, First Out (FIFO) principle, meaning the first element added to the queue is the first one to be removed.
   - Basic queue operations:
     - Enqueue Operation - Adds an element to the rear end of the queue.
     - Dequeue Operation - Removes and returns the element at the front end of the queue.

3. **Handling of queue overflow and underflow conditions:**
   - Relevant messages are displayed in case of queue overflow or underflow.

4. **Freeing Memory:**
    - The dynamically allocated memory is freed using free() to avoid memory leaks.

Algorithm:

Step 1: Initialize

i. Allocate memory for the queue q with size max using malloc.

ii. Set rear = -1, front = 0, count = 0 to indicate the queue is empty.

Step 2: Enqueue Operation

1. Check if the queue is full:
    - If count == max, it means the queue is full.
    - Print a message indicating overflow error.
2. Insert the element:
    - Increment rear by 1 (modulo addition).
    - Insert the element ele at position a[rear].
    - Increment count by 1.

Step 3: Dequeue Operation

1. Check if the queue is empty:
    - If count == 0, print a message indicating "Queue underflow".
2. Delete the element:
    - Print the deleted element, the element at q[front].
    - Increment front by 1 (modulo addition).

Step 4: Display Operation

1. Check if the queue is empty:
    - If count == 0, print "Queue is empty."
2. Display the queue elements:
    - Print ‘count’ number of elements from q[front], using modulo addition for index incrementing.

Step 5: Main Function

1. Initialize queue size:
    - Ask the user to input the initial value of max and allocate memory for the queue q using malloc.
2. Loop through the menu:
    - Print the menu and ask the user to enter a choice.
    - Based on the user's choice, call the corresponding function:
        - Choice 1: Call Enqueue() to insert an element into the queue.
        - Choice 2: Call Dequeue() to delete and display the deleted element.
        - Choice 3: Call display() to print all elements in the queue.
        - Choice 4: Exit the program and free the allocated memory.
    - If the user enters an invalid choice, print "Invalid choice."

Step 6: Terminate

- When the user chooses to exit, free the allocated memory for the queue and terminate the program.