**Ex. No: 9**

**Date:**

**IPC USING SEMAPHORES – PRODUCER, CONSUMER PROBLEM**

**Problem Statement:**

Implement a program to solve the Producer-Consumer problem using semaphores and IPC.

**Problem Description:**

The Producer-Consumer problem involves two types of processes, producers and consumers, who share a common, fixed-size buffer as a queue. Producers are responsible for producing items and adding them to the buffer, while consumers retrieve and consume items from the buffer. The challenge is to ensure that producers do not produce when the buffer is full, and consumers do not consume when the buffer is empty. Semaphores are used to synchronize access to the buffer and ensure that producers and consumers work together without conflicts.

**Algorithm:**

1. Initialize semaphores: empty, full, and mutex.
2. Create a shared buffer.
3. Implement the producer function to produce and add items to the buffer.
4. Implement the consumer function to consume items from the buffer.
5. Create producer and consumer threads.
6. Start the threads.
7. Wait for both threads to finish

**Code:**

import threading

import time

# Constants

BUFFER\_SIZE = 5

MAX\_NUMBER = 25

# Semaphores

empty = threading.Semaphore(BUFFER\_SIZE)

full = threading.Semaphore(0)

mutex = threading.Semaphore(1)

# Shared buffer

buffer = []

# Producer function

def producer():

    for item in range(1, MAX\_NUMBER + 1):  # Produce numbers from 1 to 25

        empty.acquire()  # Wait for an empty slot

        mutex.acquire()  # Obtain the mutex to access the buffer

        buffer.append(item)  # Add the item to the buffer

        print(f"Produced: {item}")

        mutex.release()  # Release the mutex

        full.release()  # Signal that the buffer is no longer empty

        time.sleep(0.1)  # Simulate some work

# Consumer function

def consumer():

    for \_ in range(MAX\_NUMBER):  # Consume a total of 25 items

        full.acquire()  # Wait for a full buffer

        mutex.acquire()  # Obtain the mutex to access the buffer

        item = buffer.pop(0)  # Consume the item from the buffer

        print(f"Consumed: {item}")

        mutex.release()  # Release the mutex

        empty.release()  # Signal that there's an empty slot in the buffer

        time.sleep(0.1)  # Simulate some work

# Create producer and consumer threads

producer\_thread = threading.Thread(target=producer)

consumer\_thread = threading.Thread(target=consumer)

# Start the threads

producer\_thread.start()

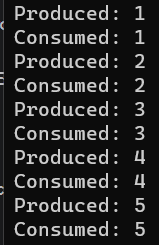
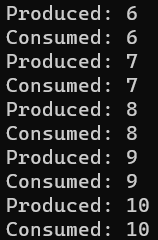
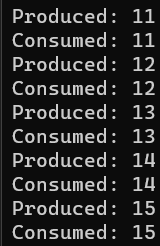
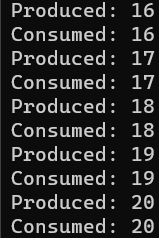
consumer\_thread.start()

# Wait for both threads to finish

producer\_thread.join()

consumer\_thread.join()

**Output:**

**   **

**Result:**

Thus, IPC using semaphores has been implemented successfully to solve producer consumer problem