**Experiment No. 7**

NAME:Omkar Khanvilkar ROLL NO: 07

CLASS: TY\_IT-B BATCH: 2

DATE OF PERFORMANCE: 26/09/2024 PRN No: 12210313

# Producer-Consumer Problem using Threads and Semaphore (with dynamic input):

## Code:

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#define BUFFER\_SIZE 5

int buffer[BUFFER\_SIZE];

int count = 0;

sem\_t empty;

sem\_t full;

pthread\_mutex\_t mutex;

void\* producer(void\* arg) {

    int id = \*((int\*)arg);

    int item;

    for (int i = 0; i < 5; i++) {

        item = rand() % 100;

        sem\_wait(&empty);

        pthread\_mutex\_lock(&mutex);

        buffer[count] = item;

        count++;

        printf("Producer %d produced item %d\n", id, item);

        pthread\_mutex\_unlock(&mutex);

        sem\_post(&full);

    }

    return NULL;

}

void\* consumer(void\* arg) {

    int id = \*((int\*)arg);

    int item;

    for (int i = 0; i < 5; i++) {

        sem\_wait(&full);

        pthread\_mutex\_lock(&mutex);

        item = buffer[count - 1];

        count--;

        printf("Consumer %d consumed item %d\n", id, item);

        pthread\_mutex\_unlock(&mutex);

        sem\_post(&empty);

    return NULL;

}

int main() {

    int numProducers, numConsumers;

    printf("Enter the number of producers: ");

    scanf("%d", &numProducers);

    printf("Enter the number of consumers: ");

    scanf("%d", &numConsumers);

    pthread\_t producers[numProducers], consumers[numConsumers];

    int producerIds[numProducers], consumerIds[numConsumers];

    sem\_init(&empty, 0, BUFFER\_SIZE);

    sem\_init(&full, 0, 0);

    pthread\_mutex\_init(&mutex, NULL);

    for (int i = 0; i < numProducers; i++) {

        producerIds[i] = i + 1;

        pthread\_create(&producers[i], NULL, producer, &producerIds[i]);

    }

    for (int i = 0; i < numConsumers; i++) {

        consumerIds[i] = i + 1;

        pthread\_create(&consumers[i], NULL, consumer, &consumerIds[i]);

    }

    for (int i = 0; i < numProducers; i++) {

        pthread\_join(producers[i], NULL);

    }

    for (int i = 0; i < numConsumers; i++) {

        pthread\_join(consumers[i], NULL);

    }

    sem\_destroy(&empty);

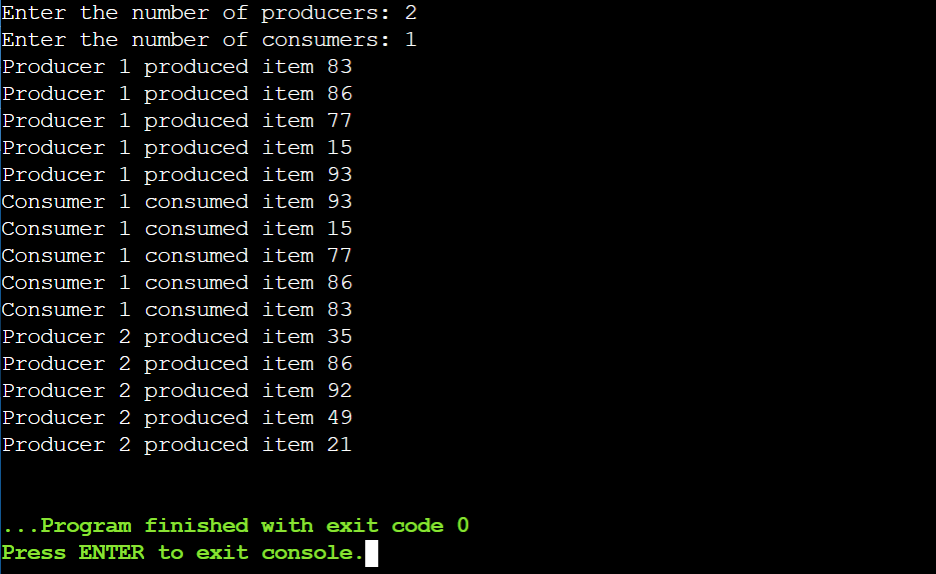
    sem\_destroy(&full);

    pthread\_mutex\_destroy(&mutex);

    return 0;

}

## Output:



# Producer-Consumer Problem using Threads and Mutex (with dynamic input):

## Code:

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#define BUFFER\_SIZE 5

int buffer[BUFFER\_SIZE];

int count = 0;

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t notFull = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_t notEmpty = PTHREAD\_COND\_INITIALIZER;

void \*producer(void \*arg)

{

    int id = \*((int \*)arg);

    int item;

    for (int i = 0; i < 5; i++)

    {

        item = rand() % 100;

        pthread\_mutex\_lock(&mutex);

        while (count == BUFFER\_SIZE)

        {

            pthread\_cond\_wait(&notFull, &mutex);

        }

        buffer[count] = item;

        count++;

        printf("Producer %d produced item %d\n", id, item);

        pthread\_cond\_signal(&notEmpty);

        pthread\_mutex\_unlock(&mutex);

    }

    return NULL;

}

void \*consumer(void \*arg)

{

    int id = \*((int \*)arg);

    int item;

    for (int i = 0; i < 5; i++)

    {

        pthread\_mutex\_lock(&mutex);

        while (count == 0)

        {

            pthread\_cond\_wait(&notEmpty, &mutex);

        }

        item = buffer[count - 1];

        count--;

        printf("Consumer %d consumed item %d\n", id, item);

        pthread\_cond\_signal(&notFull);

        pthread\_mutex\_unlock(&mutex);

    }

    return NULL;

}

int main()

{

    int numProducers, numConsumers;

    printf("Enter the number of producers: ");

    scanf("%d", &numProducers);

    printf("Enter the number of consumers: ");

    scanf("%d", &numConsumers);

    pthread\_t producers[numProducers], consumers[numConsumers];

    int producerIds[numProducers], consumerIds[numConsumers];

    for (int i = 0; i < numProducers; i++)

    {

        producerIds[i] = i + 1;

        pthread\_create(&producers[i], NULL, producer, &producerIds[i]);

    }

    for (int i = 0; i < numConsumers; i++)

    {

        consumerIds[i] = i + 1;

        pthread\_create(&consumers[i], NULL, consumer, &consumerIds[i]);

    }

    for (int i = 0; i < numProducers; i++)

    {

        pthread\_join(producers[i], NULL);

    }

    for (int i = 0; i < numConsumers; i++)

    {

        pthread\_join(consumers[i], NULL);

    }

    pthread\_mutex\_destroy(&mutex);

    pthread\_cond\_destroy(&notFull);

    pthread\_cond\_destroy(&notEmpty);

    return 0;

}

## Output:

