Experiment No. 6

NAME: Omkar Khanvilkar ROLL NO: 07

CLASS: TY_IT-B BATCH: 2

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

---

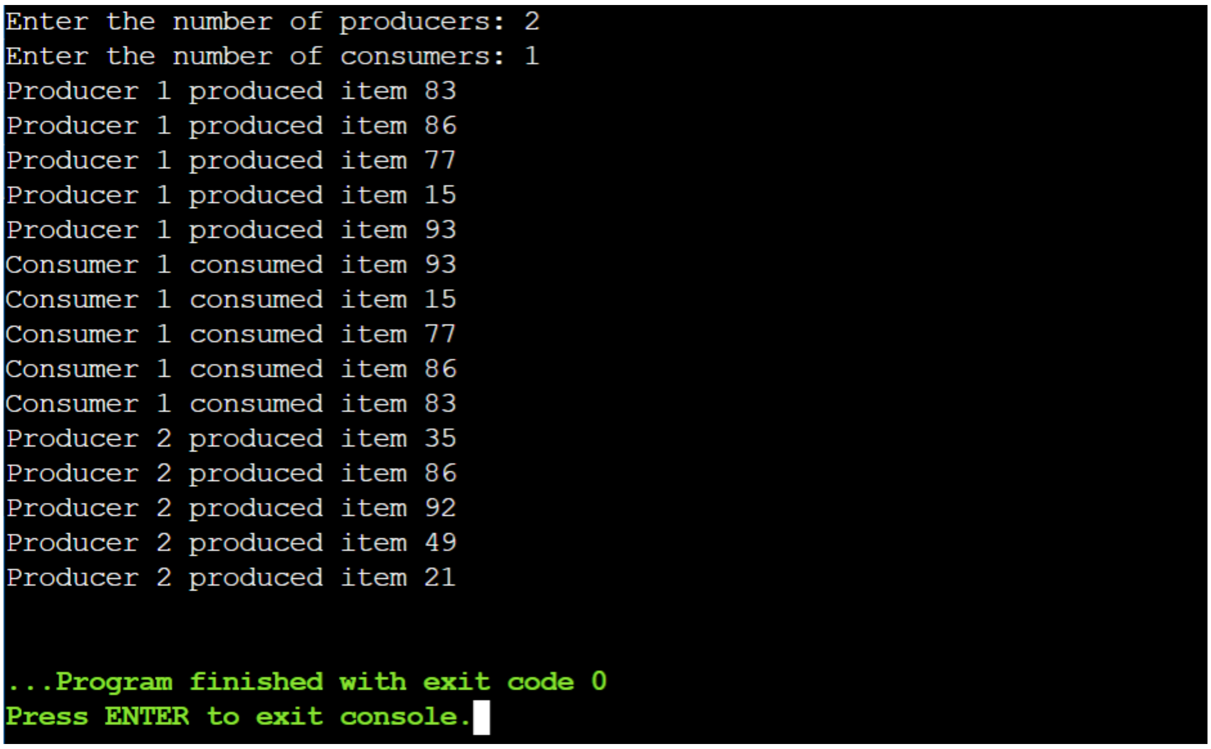
a) Producer-Consumer Problem using Threads and Semaphore (with dynamic input):

Code:

```c
#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; // Produce a random item
        sem_wait(&empty); // Wait for empty slot
        pthread_mutex_lock(&mutex); // Acquire lock to modify buffer

        buffer[count] = item;
        count++;
        printf("Producer %d produced item %d\n", id, item);

        pthread_mutex_unlock(&mutex); // Release lock
        sem_post(&full); // Signal that buffer has a new item
    }

    return NULL;
}
```

```c
void* consumer(void* arg) {
    int id = *((int*)arg);
    int item;

    for (int i = 0; i < 5; i++) {
        sem_wait(&full); // Wait for a full slot
        pthread_mutex_lock(&mutex); // Acquire lock to modify buffer

        item = buffer[count - 1];
        count--;
        printf("Consumer %d consumed item %d\n", id, item);

        pthread_mutex_unlock(&mutex); // Release lock
        sem_post(&empty); // Signal that buffer has an empty slot
    }

    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:



```
Enter the number of producers: 2
Enter the number of consumers: 1
Producer 1 produced item 83
Producer 1 produced item 86
Producer 1 produced item 77
Producer 1 produced item 15
Producer 1 produced item 93
Consumer 1 consumed item 93
Consumer 1 consumed item 15
Consumer 1 consumed item 77
Consumer 1 consumed item 86
Consumer 1 consumed item 83
Producer 2 produced item 35
Producer 2 produced item 86
Producer 2 produced item 92
Producer 2 produced item 49
Producer 2 produced item 21


...Program finished with exit code 0
Press ENTER to exit console.
```

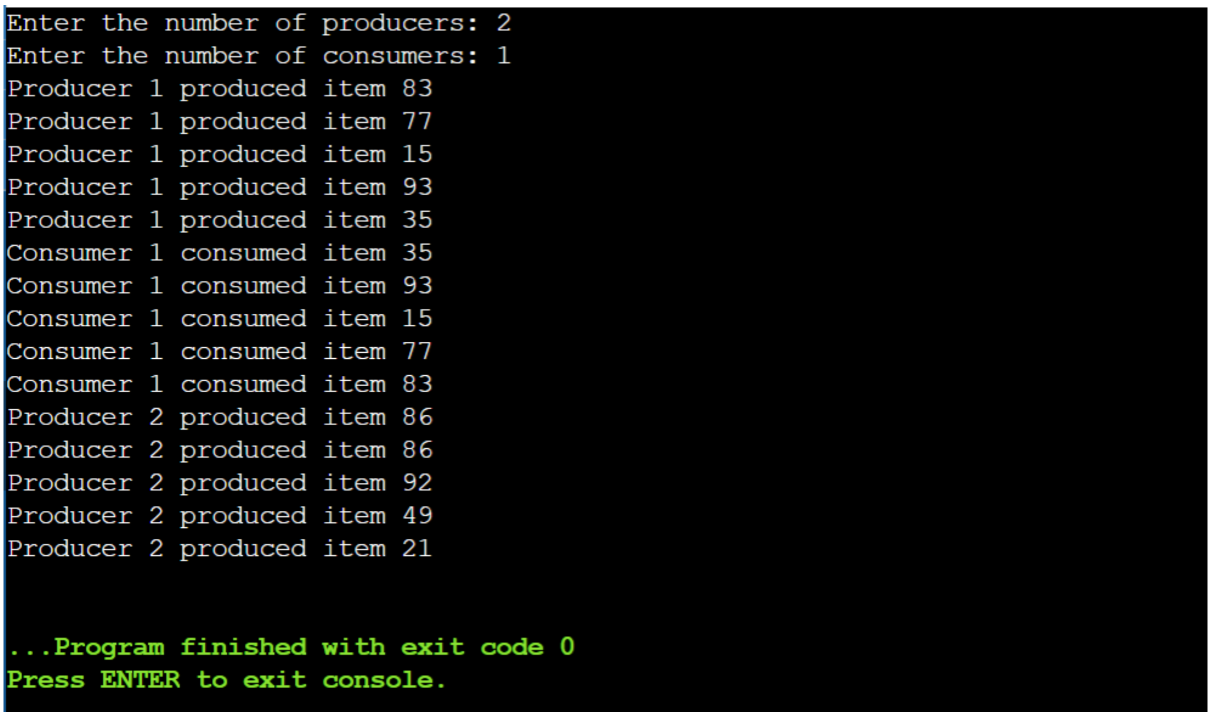
b) Producer-Consumer Problem using Threads and Mutex (with dynamic input):

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; // Produce a random item
        pthread_mutex_lock(&mutex);

        while (count == BUFFER_SIZE)
        {
            pthread_cond_wait(&notFull, &mutex); // Wait for space in
buffer
        }

        buffer[count] = item;
        count++;
        printf("Producer %d produced item %d\n", id, item);

        pthread_cond_signal(&notEmpty); // Signal that buffer has new
items
        pthread_mutex_unlock(&mutex);
    }

    return NULL;
}

void *consumer(void *arg)
```

```c
{
    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); // Wait for items in
buffer
        }

        item = buffer[count - 1];
        count--;
        printf("Consumer %d consumed item %d\n", id, item);

        pthread_cond_signal(&notFull); // Signal that buffer has empty
space
        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:



```
Enter the number of producers: 2
Enter the number of consumers: 1
Producer 1 produced item 83
Producer 1 produced item 77
Producer 1 produced item 15
Producer 1 produced item 93
Producer 1 produced item 35
Consumer 1 consumed item 35
Consumer 1 consumed item 93
Consumer 1 consumed item 15
Consumer 1 consumed item 77
Consumer 1 consumed item 83
Producer 2 produced item 86
Producer 2 produced item 86
Producer 2 produced item 92
Producer 2 produced item 49
Producer 2 produced item 21


...Program finished with exit code 0
Press ENTER to exit console.
```