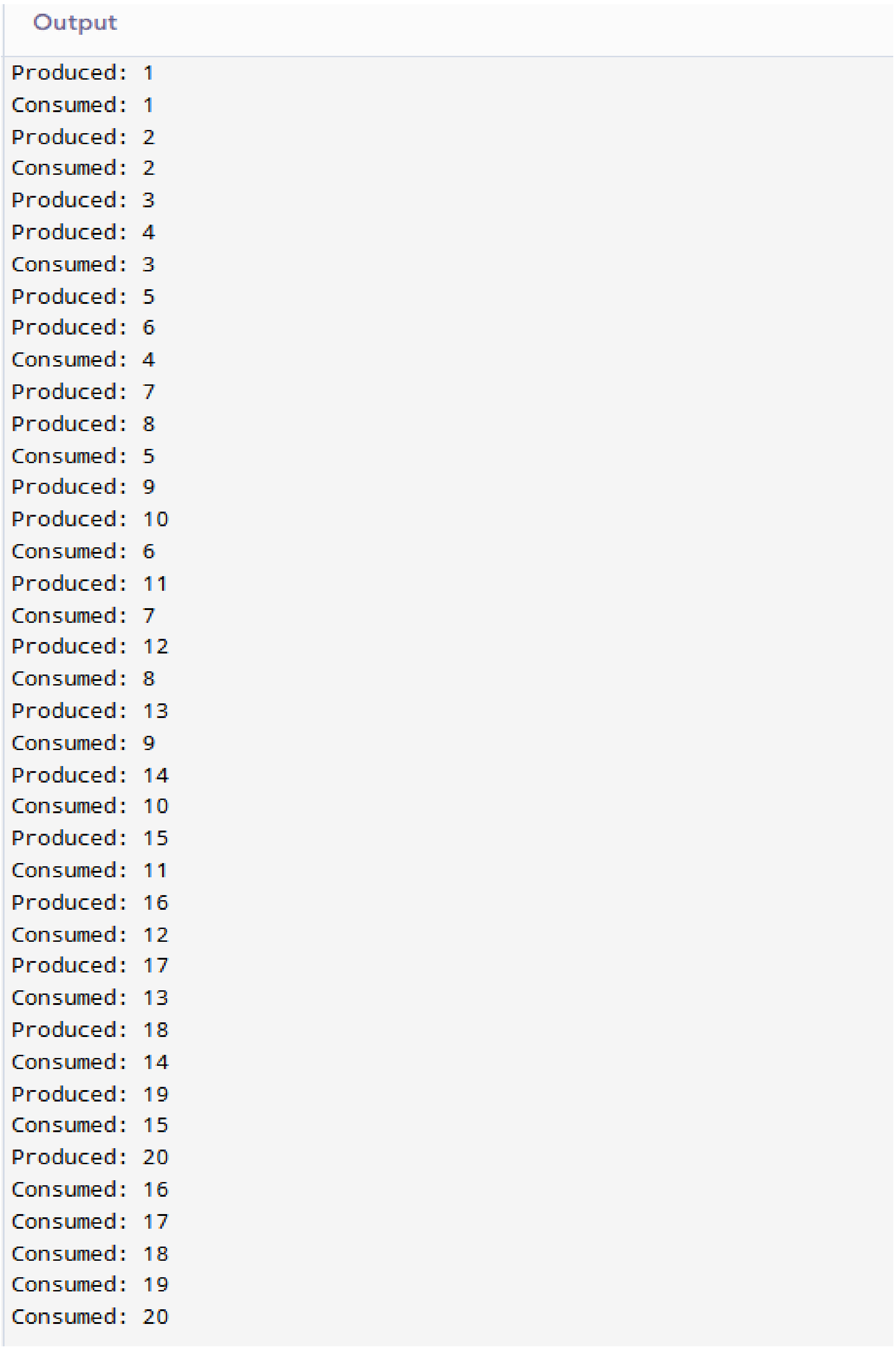
Name: Prem Rajput PRN-23062701242053

Experiment No. 7
Implementation of Classical Synchronization Problems

Producer-Consumer problem.

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define SIZE 5
#define MAX_ITEMS 20
int buffer[SIZE];
int in = 0, out = 0;
int produced_count = 0, consumed_count = 0;
sem_t empty, full;
pthread_mutex_t mutex;
void* producer(void* arg) {
    int item = 1;
    while (produced_count < MAX_ITEMS) {
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = item;
        printf("Produced: %d\n", item);
        item++;
        in = (in + 1) % SIZE;
        produced_count++;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
        sleep(1);
    }
    return NULL;
}
void* consumer(void* arg) {
    while (consumed_count < MAX_ITEMS) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
        printf("Consumed: %d\n", item);
        out = (out + 1) % SIZE;
        consumed_count++;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
        sleep(2);
    }
    return NULL;
}
int main() {
    pthread_t p, c;
    sem_init(&empty, 0, SIZE);
    sem_init(&full, 0, 0);
    pthread_mutex_init(&mutex, NULL);
    pthread_create(&p, NULL, producer, NULL);
    pthread_create(&c, NULL, consumer, NULL);
    pthread_join(p, NULL);
    pthread_join(c, NULL);
    sem_destroy(&empty);
    sem_destroy(&full);
    pthread_mutex_destroy(&mutex);
    return 0;
}
```



Output

```
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Produced: 4
Consumed: 3
Produced: 5
Produced: 6
Consumed: 4
Produced: 7
Produced: 8
Consumed: 5
Produced: 9
Produced: 10
Consumed: 6
Produced: 11
Consumed: 7
Produced: 12
Consumed: 8
Produced: 13
Consumed: 9
Produced: 14
Consumed: 10
Produced: 15
Consumed: 11
Produced: 16
Consumed: 12
Produced: 17
Consumed: 13
Produced: 18
Consumed: 14
Produced: 19
Consumed: 15
Produced: 20
Consumed: 16
Consumed: 17
Consumed: 18
Consumed: 19
Consumed: 20
```

Name: Prem Rajput PRN-23062701242053

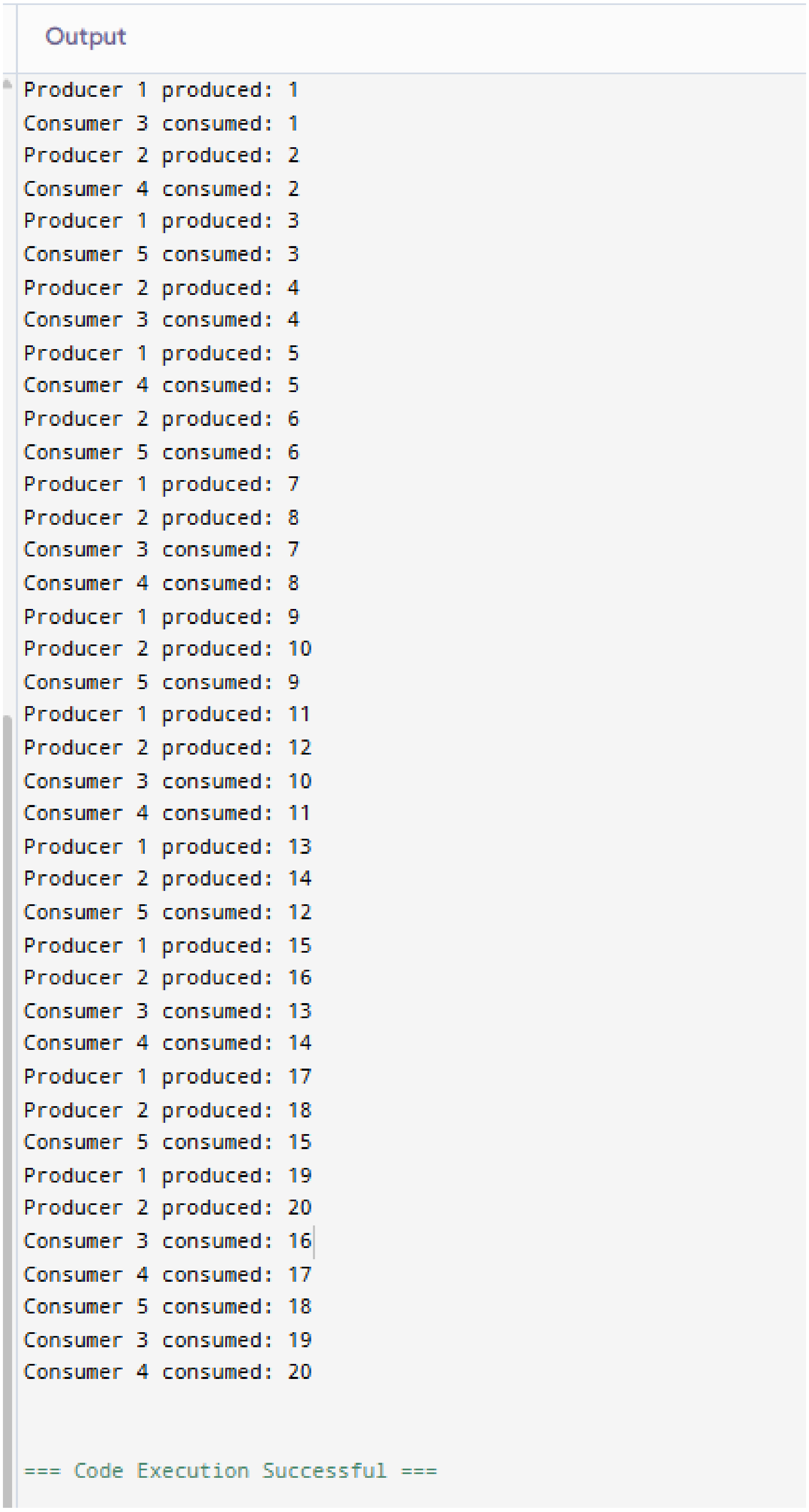
```
//reader writer problem
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define SIZE 5
#define MAX_ITEMS 20

int buffer[SIZE];
int in = 0, out = 0;
int produced_count = 0, consumed_count = 0;
sem_t empty, full;
pthread_mutex_t mutex;

void* producer(void* arg) {
    int id = *((int*)arg);
    while (produced_count < MAX_ITEMS) {
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = produced_count + 1;
        printf("Producer %d produced: %d\n", id, buffer[in]);
        in = (in + 1) % SIZE;
        produced_count++;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
        sleep(1);
    }
    return NULL;
}
void* consumer(void* arg) {
    int id = *((int*)arg);
    while (consumed_count < MAX_ITEMS) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
        printf("Consumer %d consumed: %d\n", id, item);
        out = (out + 1) % SIZE;
        consumed_count++;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
        sleep(2);
 }
    return NULL;
}
int main() {
    pthread_t producers[2], consumers[3];
    int ids[5] = {1, 2, 3, 4, 5};
    int i;
    sem_init(&empty, 0, SIZE);
    sem_init(&full, 0, 0);
    pthread_mutex_init(&mutex, NULL);
    for (i = 0; i < 2; i++) {
        pthread_create(&producers[i], NULL, producer, &ids[i]);
    }
    for (i = 0; i < 3; i++) {
        pthread_create(&consumers[i], NULL, consumer, &ids[i + 2]);
    }
    for (i = 0; i < 2; i++) {
        pthread_join(producers[i], NULL);
    }
    for (i = 0; i < 3; i++) {
        pthread_join(consumers[i], NULL);
    }
    sem_destroy(&empty);
   sem_destroy(&full);
   pthread_mutex_destroy(&mutex);
   return 0;
}
```

Name: Prem Rajput PRN-23062701242053



Output

```
Producer 1 produced: 1
Consumer 3 consumed: 1
Producer 2 produced: 2
Consumer 4 consumed: 2
Producer 1 produced: 3
Consumer 5 consumed: 3
Producer 2 produced: 4
Consumer 3 consumed: 4
Producer 1 produced: 5
Consumer 4 consumed: 5
Producer 2 produced: 6
Consumer 5 consumed: 6
Producer 1 produced: 7
Producer 2 produced: 8
Consumer 3 consumed: 7
Consumer 4 consumed: 8
Producer 1 produced: 9
Producer 2 produced: 10
Consumer 5 consumed: 9
Producer 1 produced: 11
Producer 2 produced: 12
Consumer 3 consumed: 10
Consumer 4 consumed: 11
Producer 1 produced: 13
Producer 2 produced: 14
Consumer 5 consumed: 12
Producer 1 produced: 15
Producer 2 produced: 16
Consumer 3 consumed: 13
Consumer 4 consumed: 14
Producer 1 produced: 17
Producer 2 produced: 18
Consumer 5 consumed: 15
Producer 1 produced: 19
Producer 2 produced: 20
Consumer 3 consumed: 16
Consumer 4 consumed: 17
Consumer 5 consumed: 18
Consumer 3 consumed: 19
Consumer 4 consumed: 20


=== Code Execution Successful ===
```