NATIONAL TEXTILE

UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

SUBMITTED BY:

Eman Faisal 23-NTU-CS-1149

SECTION SE: 5th (A)

Operating System-Lab plan 10

SUBMITTED TO:

Sir Nasir Mahmood

SUBMISSION DATE: 28/11/25

Task Code:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define BUFFER_SIZE 5
int buffer[BUFFER_SIZE];
int in = 0; // Producer index
int out = 0; // Consumer index
sem_t empty; // Counts empty slots
sem_t full; // Counts full slots
pthread_mutex_t mutex;
void* producer(void* arg) {
int id = *(int*)arg;
for(int i = 0; i < 3; i++) { // Each producer makes 3 items
int item = id * 100 + i;
// TODO: Wait for empty slot
sem_wait(&empty);
// TODO: Lock the buffer
pthread_mutex_lock(&mutex);
// Add item to buffer
buffer[in] = item;
printf("Producer %d produced item %d at position %d\n",
id, item, in);
in = (in + 1) % BUFFER_SIZE;
// TODO: Unlock the buffer
pthread_mutex_unlock(&mutex);
// TODO: Signal that buffer has a full slot
sem_post(&full);
sleep(1);
}
return NULL;
 }
void* consumer(void* arg) {
int id = *(int*)arg;
for(int i = 0; i < 3; i++) {
// TODO: Students complete this similar to producer
sem_wait(&full);
pthread_mutex_lock(&mutex);
int item = buffer[out];
printf("Consumer %d consumed item %d from position %d\n",
id, item, out);
```

```
 out = (out + 1) % BUFFER_SIZE;
 pthread_mutex_unlock(&mutex);
 sem_post(&empty);
 sleep(2); // Consumers are slower
 }
 return NULL;
 }
 int main() {
pthread_t prod[2], cons[2];
 int ids[2] = {1, 2};
 // Initialize semaphores
 sem_init(&empty, 0, BUFFER_SIZE); // All slots empty initially
 sem_init(&full, 0, 0);
 pthread_mutex_init(&mutex, NULL);
 // No slots full initially
 // Create producers and consumers
 for(int i = 0; i < 2; i++) {
 pthread_create(&prod[i], NULL, producer, &ids[i]);
 pthread_create(&cons[i], NULL, consumer, &ids[i]);
 }
 // Wait for completion
 for(int i = 0; i < 2; i++) {
 pthread_join(prod[i], NULL);
 pthread_join(cons[i], NULL);
 }
 // Cleanup
 sem_destroy(&empty);
 sem_destroy(&full);
 pthread_mutex_destroy(&mutex);
 return 0;
}
```

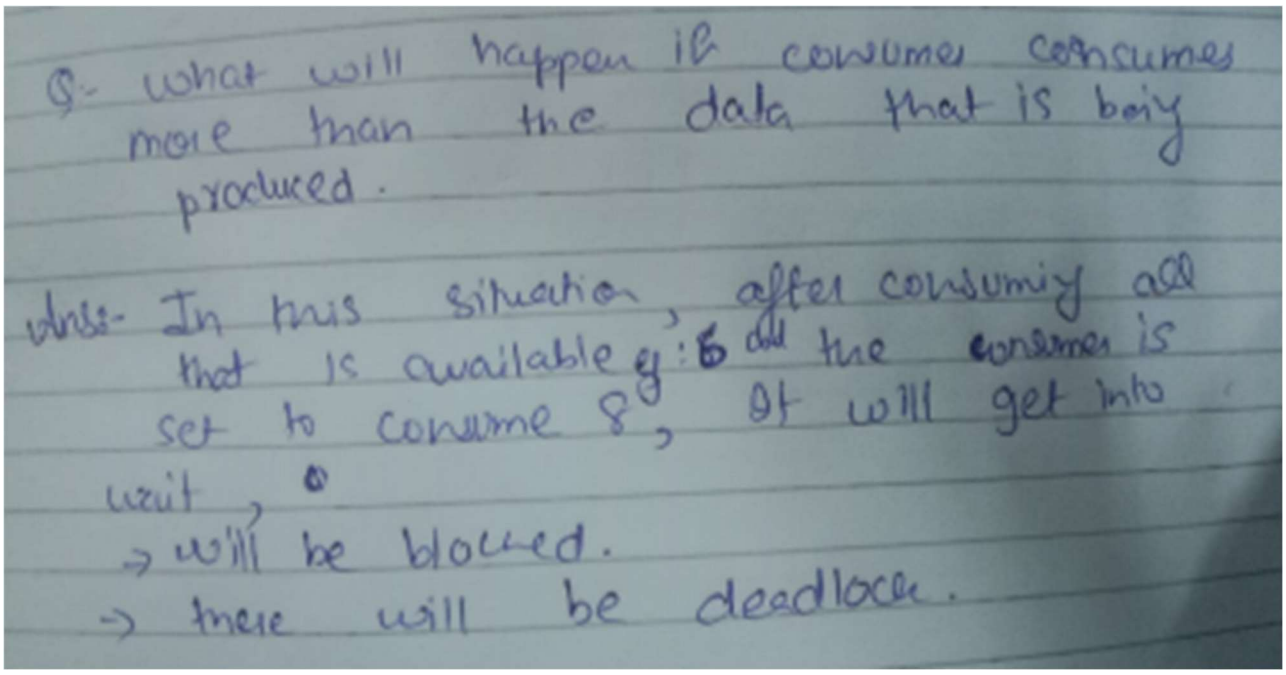
Output:

```
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/labplan10# ./a.out
Producer 1 produced item 100 at position 0
Consumer 1 consumed item 100 from position 0
Producer 2 produced item 200 at position 1
Consumer 2 consumed item 200 from position 1
Producer 1 produced item 101 at position 2
Producer 2 produced item 201 at position 3
Consumer 1 consumed item 101 from position 2
Consumer 2 consumed item 201 from position 3
Producer 1 produced item 102 at position 4
Producer 2 produced item 202 at position 0
Consumer 1 consumed item 102 from position 4
Consumer 2 consumed item 202 from position 0
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/labplan10#
```

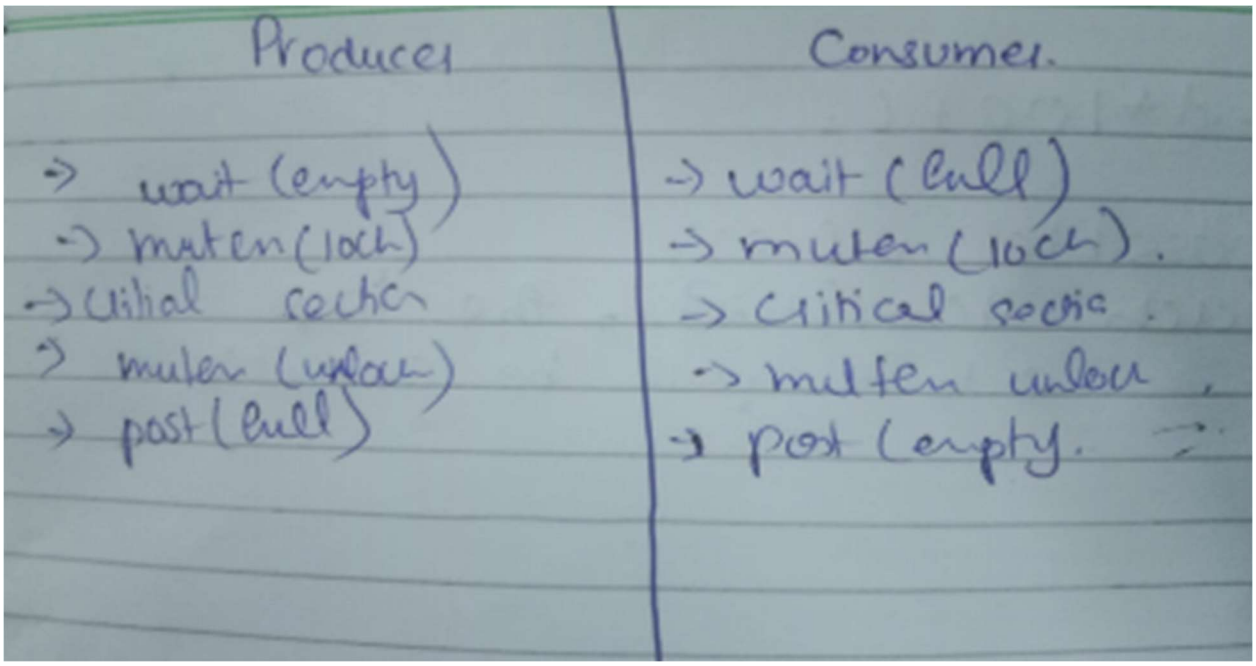
Technical Description:

- There are total **4 threads** In this program 2 for producer, 2 for consumer.
- There are **2 semaphores**. Empty keeps track of empty places. Full keeps track of full places.
- **In producer,** we use wait(&empty) .This subtracts 1 from the number of slots that are empty, indicating that there is one less empty space now.Then,we use post(&full) after critical section ,it adds 1 to the full slots number.
- **In consumer,** we use wait(&full) .This adds 1 to the number of full slots, indicating that there is one more fullspace now.Then, we use post(&empty) after critical section ,it subtracts 1 from the empty slots number.
- **Item=id*100+1:**
  For each id, where id={1,2},there will be loop from 0 to 2, each loop will create 3 values.So, there will be **total of 6 values** produces.
  For id=1, possible values: 100,101,102
  For id=2, possible values: 200,201,202

Q.What will happen if consumer is set to consume more than the number of data items that are produced?



Q.Difference in producer consumer implementation steps:



Q.How many possible values of item can be produced referring to this line of code: Item=id*100+1?

For each id, where id={1,2},there will be loop from 0 to 2, each loop will create 3 values.So, there will be **total of 6 values** produces.
For id=1, possible values: 100,101,102.
For id=2,possible values:200,201,202.