NATIONAL TEXTILE

UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Subject

Operating System

SUBMITTED BY:

Fatima Waseem: 23-NTU-CS1155

SECTION SE: 5th (A)

SUBMITTED TO:

Sir Nasir

Lab 10

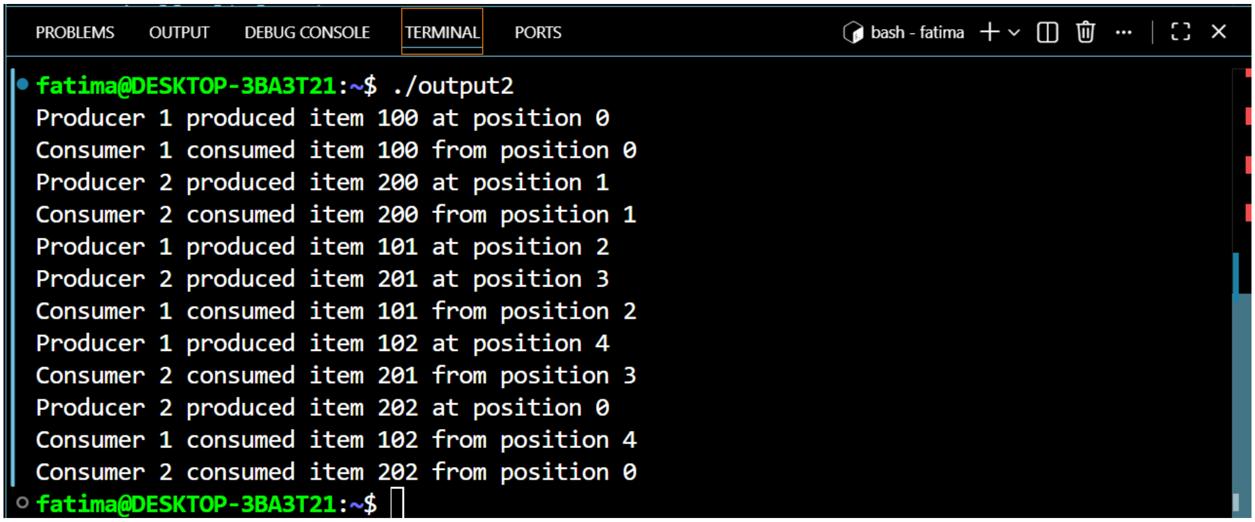
Task2 Producer consumer problem:

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;
```

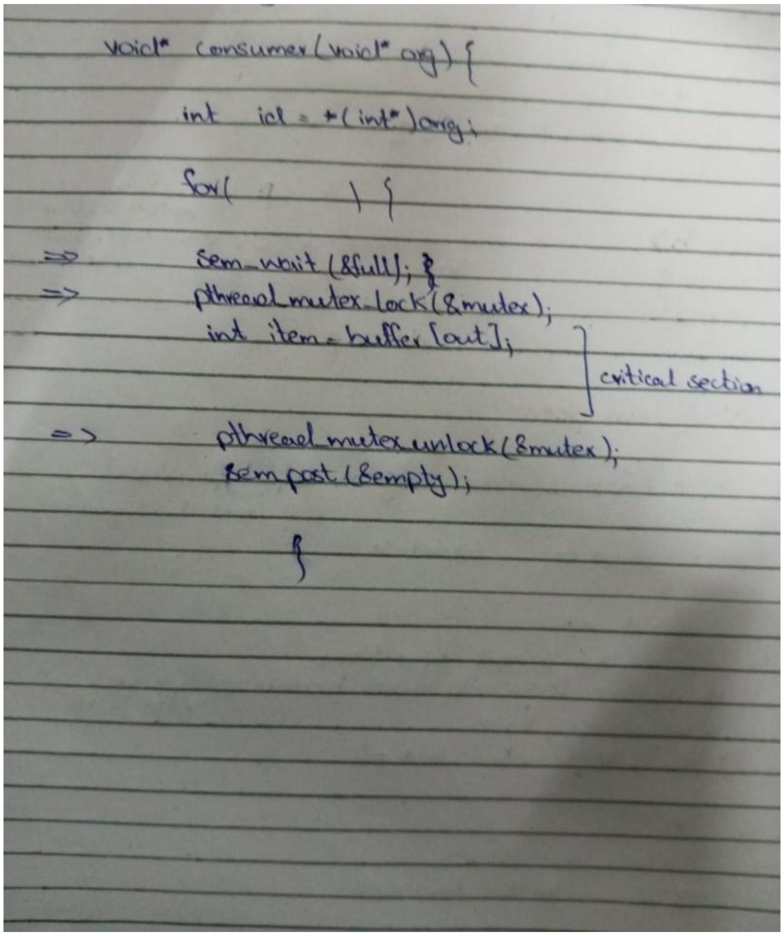
```
}
void* consumer(void* arg) {
int id = *(int*)arg;
for(int i = 0; i < 3; i++) {
// TODO: Students complete this similar to producer
sem_wait(&full);
pthread_mutex_lock(&mutex);
int item = buffer[out];
printf("Consumer %d consumed item %d from position %d\n",
id, item, out);
out = (out + 1) % BUFFER_SIZE;
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); // No slots full initially
pthread_mutex_init(&mutex, NULL);
// 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;
}

// Here we have two producer threads and two consumer threads
// Each producer is making 3 slots of items and we have 2
producer so we have total 6 slots of items that is producing
the producer
//then we decrement by sem_wait(&empty) and showing that one
slot is taken form the buffer

//then we will secure the critical section by mutex lock and
unlock
//sem_post(&full) Signal that buffer has a full slot

// Each consumer is consuming 3 slots of items and we have 2
consumers so we have total 6 slots of items that is consuming
the consumer
//then we decrement by sem_wait(&full) and showing that one
item is taken from the buffer
//then we will secure the critical section by mutex lock and
unlock
//sem_post(&empty) Signal that buffer has a empty slot
```

Output:



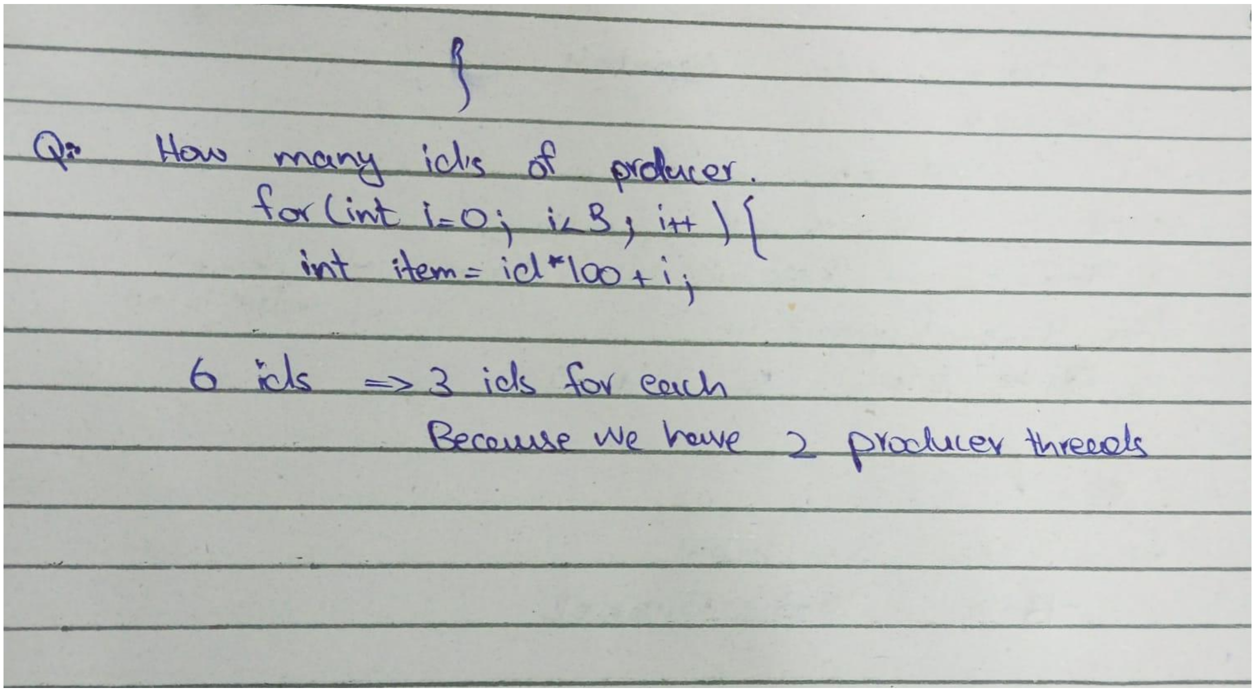
```
fatima@DESKTOP-3BA3T21:~$ ./output2
Producer 1 produced item 100 at position 0
Consumer 1 consumed item 100 from position 0
Producer 2 produced item 200 at position 1
Consumer 2 consumed item 200 from position 1
Producer 1 produced item 101 at position 2
Producer 2 produced item 201 at position 3
Consumer 1 consumed item 101 from position 2
Producer 1 produced item 102 at position 4
Consumer 2 consumed item 201 from position 3
Producer 2 produced item 202 at position 0
Consumer 1 consumed item 102 from position 4
Consumer 2 consumed item 202 from position 0
fatima@DESKTOP-3BA3T21:~$
```

Consumer steps:



```
void* consumer(void* arg) {
    int id = *(int*)arg;
    for(        ) {
=>      sem_wait(&full); }
=>      pthread_mutex_lock(&mutex);
        int item = buffer[out];          ]
                                         ] critical section
=>      pthread_mutex_unlock(&mutex);
        sem_post(&empty);

        }
```

How many ids producer will produce?



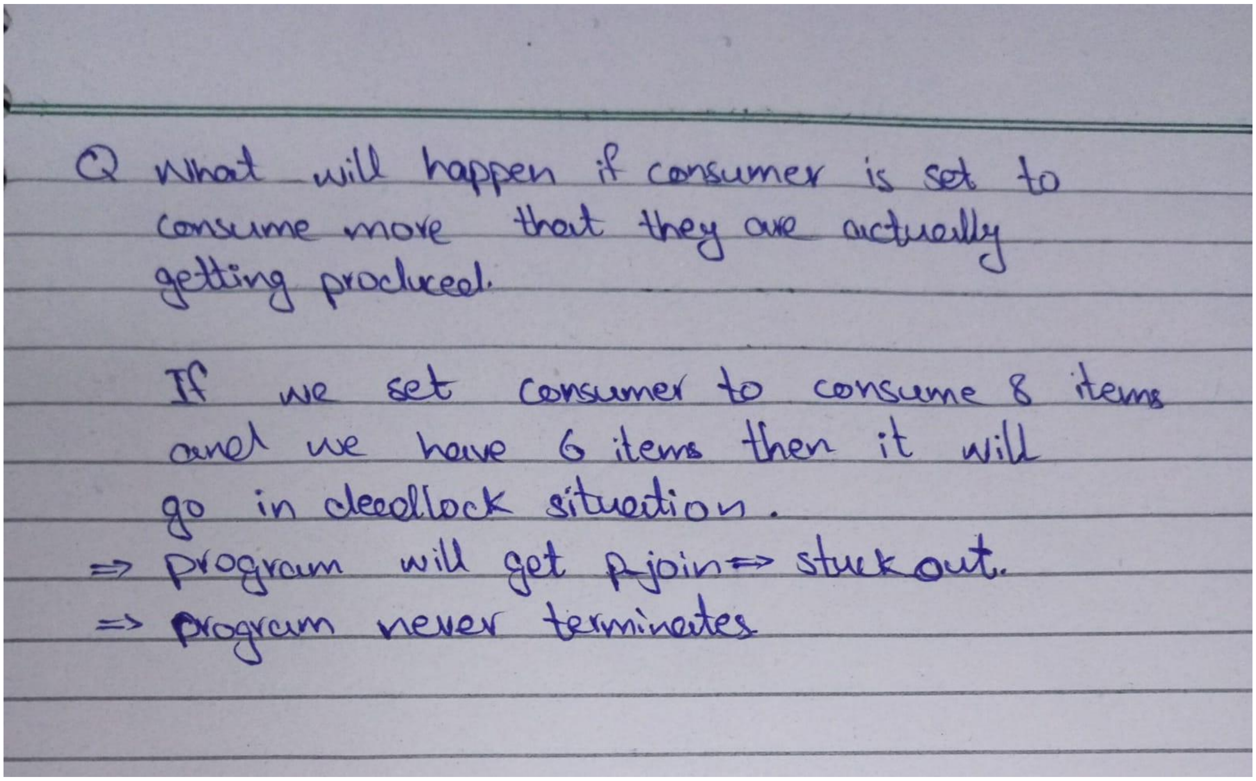
}

Q: How many ids of producer.

```
for(int i=0; i<3; i++){
    int item= id*100+i;
```

6 ids => 3 ids for each
Because we have 2 producer threads

Q: What will happen if consumer is set to consume 8 items and we have 6 items then it will go in deadlock situation?



Q What will happen if consumer is set to consume more that they are actually getting produced.

If we set consumer to consume 8 items and we have 6 items then it will go in deadlock situation.

=> program will get ~~p~~join => stuck out.

=> program never terminates