SRN: PES1UG19EC326

NAME: Tanya Chanchalani

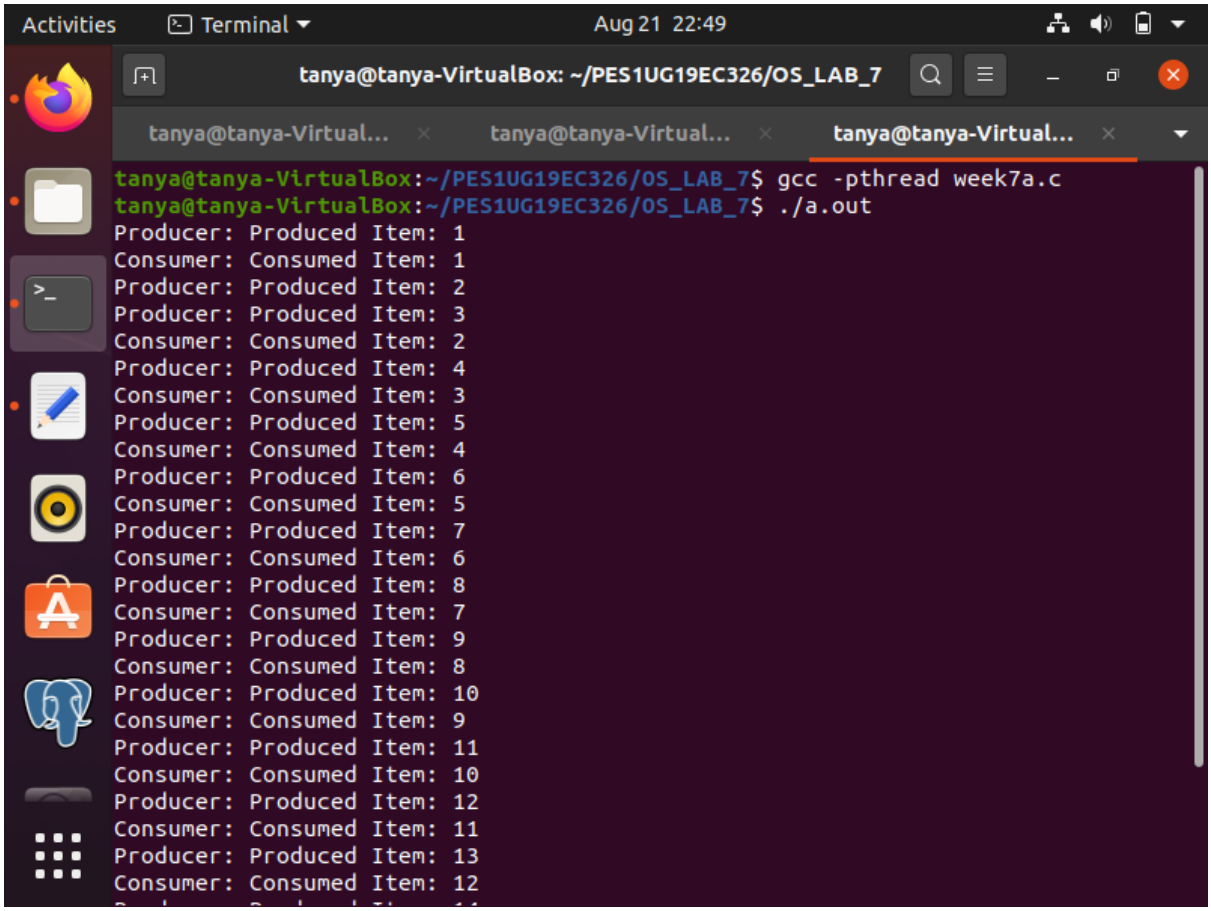
WEEK: 7

SUBJECT: OS

//SHOWING RACE CONDITION

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <pthread.h>
#define BUFFER_SIZE 50
int start = 0, end = 0;
int *buffer;
int item = 0;
void *producer()
{
   while (true)
   {
      item += 1;
      printf("Producer: Produced Item: %d\n", item);
      sleep(1);
      while (((start + 1) % BUFFER_SIZE) == end)
         ;
      buffer[start] = item;
      start = (start + 1) % BUFFER_SIZE;
   }
}
```

```
void *consumer()
{
   while (true)
   {
      while (start == end)
         ;
      int consumed = buffer[end];
      printf("Consumer: Consumed Item: %d\n", consumed);
      sleep(1);
      end = (end + 1) % BUFFER_SIZE;
   }
}
int main()
{
   buffer = (int *)malloc(sizeof(int) * BUFFER_SIZE);
   pthread_t producer_thread, consumer_thread;
   pthread_create(&producer_thread, NULL, producer, NULL);
   pthread_create(&consumer_thread, NULL, consumer, NULL);
   pthread_join(producer_thread, NULL);
   pthread_join(consumer_thread, NULL);
   free(buffer);
   return 0;
}
```



//WITHOUT RACE CONDITION

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <pthread.h>
#define BUFFER_SIZE 50
void *producer();
void *consumer();
int *buffer;
int start = 0, end = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

```
pthread_mutex_t empty = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t full = PTHREAD_MUTEX_INITIALIZER;
void *producer()
{
   int item = 0;
   while (true)
   {
      pthread_mutex_lock(&empty);
      pthread_mutex_lock(&mutex);
      item += 1;
      printf("Producer: Produced Item: %d\n", item);
      buffer[start] = item;
      pthread_mutex_unlock(&mutex);
      pthread_mutex_unlock(&full);
      start = (start + 1) % BUFFER_SIZE;
   }
}
void *consumer()
{
   while (true)
   {
      pthread_mutex_lock(&full);
      pthread_mutex_lock(&mutex);
      int consumed = buffer[end];
      printf("Consumer: Consumed Item: %d\n", consumed);
      sleep(1);
      end = (end + 1) % BUFFER_SIZE;
      pthread_mutex_unlock(&mutex);
      pthread_mutex_unlock(&empty);
   }
}
```

```
int main()
{
    buffer = (int *)malloc(sizeof(int) * BUFFER_SIZE);
    pthread_t producer_thread, consumer_thread;
    pthread_create(&producer_thread, NULL, producer, NULL);
    sleep(1);
    pthread_create(&consumer_thread, NULL, consumer, NULL);
    pthread_join(producer_thread, NULL);
    pthread_join(consumer_thread, NULL);
    free(buffer);
    return 0;
}
```