B.BHANUTEJA REDDY-192325016

29.Write a C program to simulate the solution of Classical Process Synchronization Problem

AIM

To simulate the solution of the classical process synchronization problem, such as the Producer-Consumer problem, using C programming.

ALGORITHM

1. **Start**
2. Initialize shared resources such as a buffer, semaphores, and counters.
3. Define two functions:
    - **Producer:** Produces an item and adds it to the buffer if space is available.
    - **Consumer:** Removes an item from the buffer if items are available.
4. Use semaphores to ensure mutual exclusion and synchronization between producer and consumer.
5. Implement an infinite loop for the producer and consumer to operate concurrently.
6. Display the actions of the producer and consumer.
7. Stop the program when needed (manually or after a specific condition).

PROCEDURE

1. Include necessary libraries for threading and synchronization (pthread.h and semaphore.h).
2. Initialize semaphores for synchronization:
    - empty: Tracks the number of empty slots in the buffer.
    - full: Tracks the number of filled slots in the buffer.
    - mutex: Ensures mutual exclusion while accessing the buffer.
3. Define the buffer and associated variables.
4. Implement producer and consumer functions with synchronization mechanisms.
5. Create threads for producer and consumer.
6. Use pthread_join() to wait for threads to complete execution.

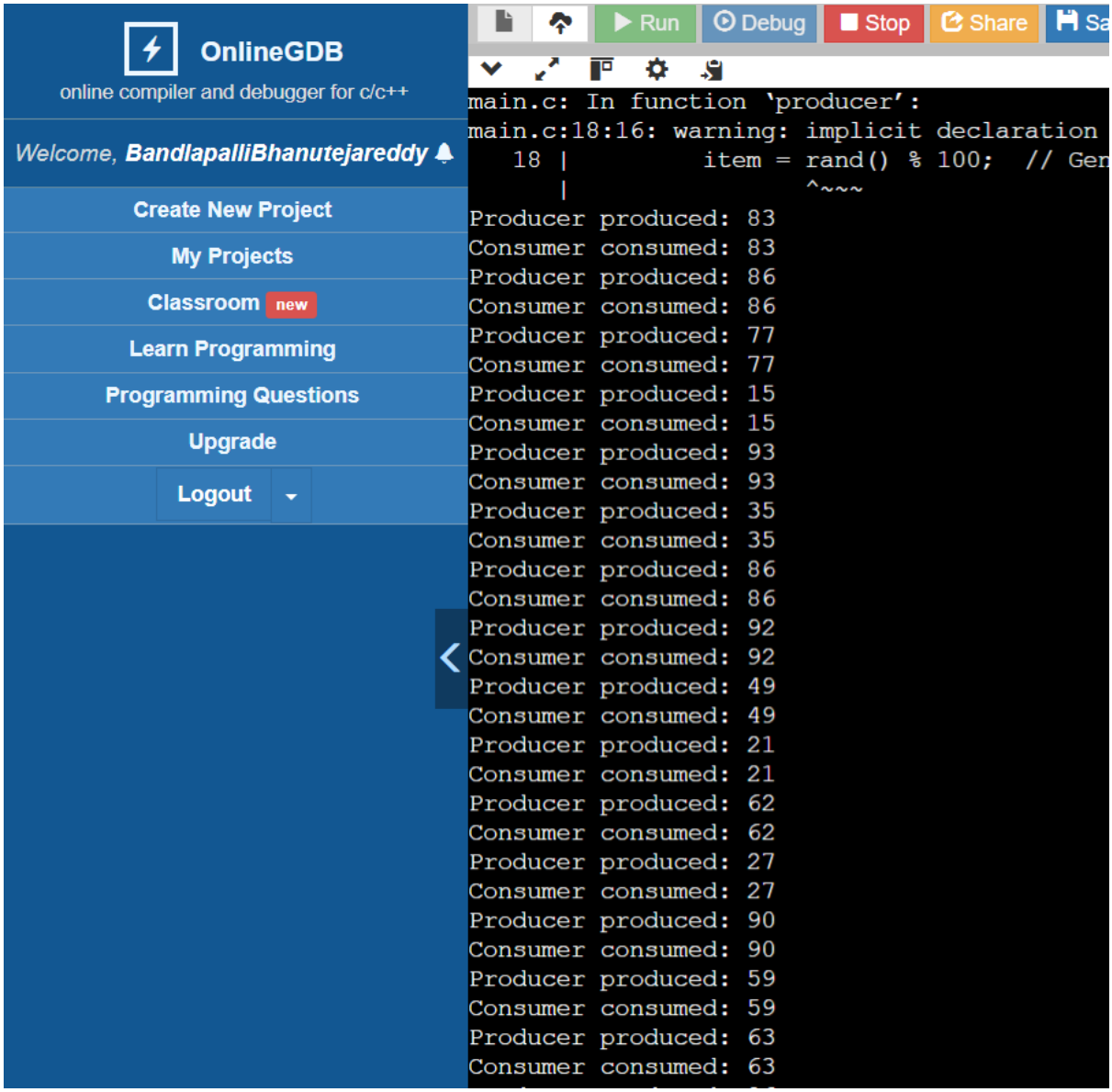
CODE:

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

#define BUFFER_SIZE 5

int buffer[BUFFER_SIZE];
int in = 0, out = 0;

sem_t empty;  // Semaphore for empty slots
sem_t full;   // Semaphore for full slots
pthread_mutex_t mutex; // Mutex for mutual exclusion

void *producer(void *arg) {
   int item;
   while (1) {
      item = rand() % 100;  // Generate a random item
      sem_wait(&empty);     // Wait for empty slot
      pthread_mutex_lock(&mutex); // Lock the buffer

      buffer[in] = item;
      printf("Producer produced: %d\n", item);
      in = (in + 1) % BUFFER_SIZE;

      pthread_mutex_unlock(&mutex); // Unlock the buffer
```

```
        sem_post(&full);      // Signal that a full slot is available

        sleep(1); // Simulate production time
    }
}

void *consumer(void *arg) {
    int item;
    while (1) {
        sem_wait(&full);      // Wait for a full slot
        pthread_mutex_lock(&mutex); // Lock the buffer

        item = buffer[out];
        printf("Consumer consumed: %d\n", item);
        out = (out + 1) % BUFFER_SIZE;

        pthread_mutex_unlock(&mutex); // Unlock the buffer
        sem_post(&empty);     // Signal that an empty slot is available

        sleep(1); // Simulate consumption time
    }
}

int main() {
    pthread_t prod, cons;

    sem_init(&empty, 0, BUFFER_SIZE); // Initialize empty slots
```

```
    sem_init(&full, 0, 0);          // Initialize full slots
    pthread_mutex_init(&mutex, NULL); // Initialize mutex

    pthread_create(&prod, NULL, producer, NULL);
    pthread_create(&cons, NULL, consumer, NULL);

    pthread_join(prod, NULL);
    pthread_join(cons, NULL);

    sem_destroy(&empty);
    sem_destroy(&full);
    pthread_mutex_destroy(&mutex);

    return 0;
}
```

OUTPUT:


OnlineGDB
online compiler and debugger for c/c++
Welcome, BandlapalliBhanutejareddy
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Run
Debug
Stop
Share
main.c: In function 'producer':
main.c:18:16: warning: implicit declaration
18 | item = rand() % 100; // Gen
| ^~~~
Producer produced: 83
Consumer consumed: 83
Producer produced: 86
Consumer consumed: 86
Producer produced: 77
Consumer consumed: 77
Producer produced: 15
Consumer consumed: 15
Producer produced: 93
Consumer consumed: 93
Producer produced: 35
Consumer consumed: 35
Producer produced: 86
Consumer consumed: 86
Producer produced: 92
Consumer consumed: 92
Producer produced: 49
Consumer consumed: 49
Producer produced: 21
Consumer consumed: 21
Producer produced: 62
Consumer consumed: 62
Producer produced: 27
Consumer consumed: 27
Producer produced: 90
Consumer consumed: 90
Producer produced: 59
Consumer consumed: 59
Producer produced: 63
Consumer consumed: 63