Operating System – CS23431

Ex 8	**Producer Consumer Using Semaphores**
Name: B M Madhumitha	
Reg No: 230701168	

Aim: To write a program to implement solution to producer consumer problem using semaphores.

Algorithm:

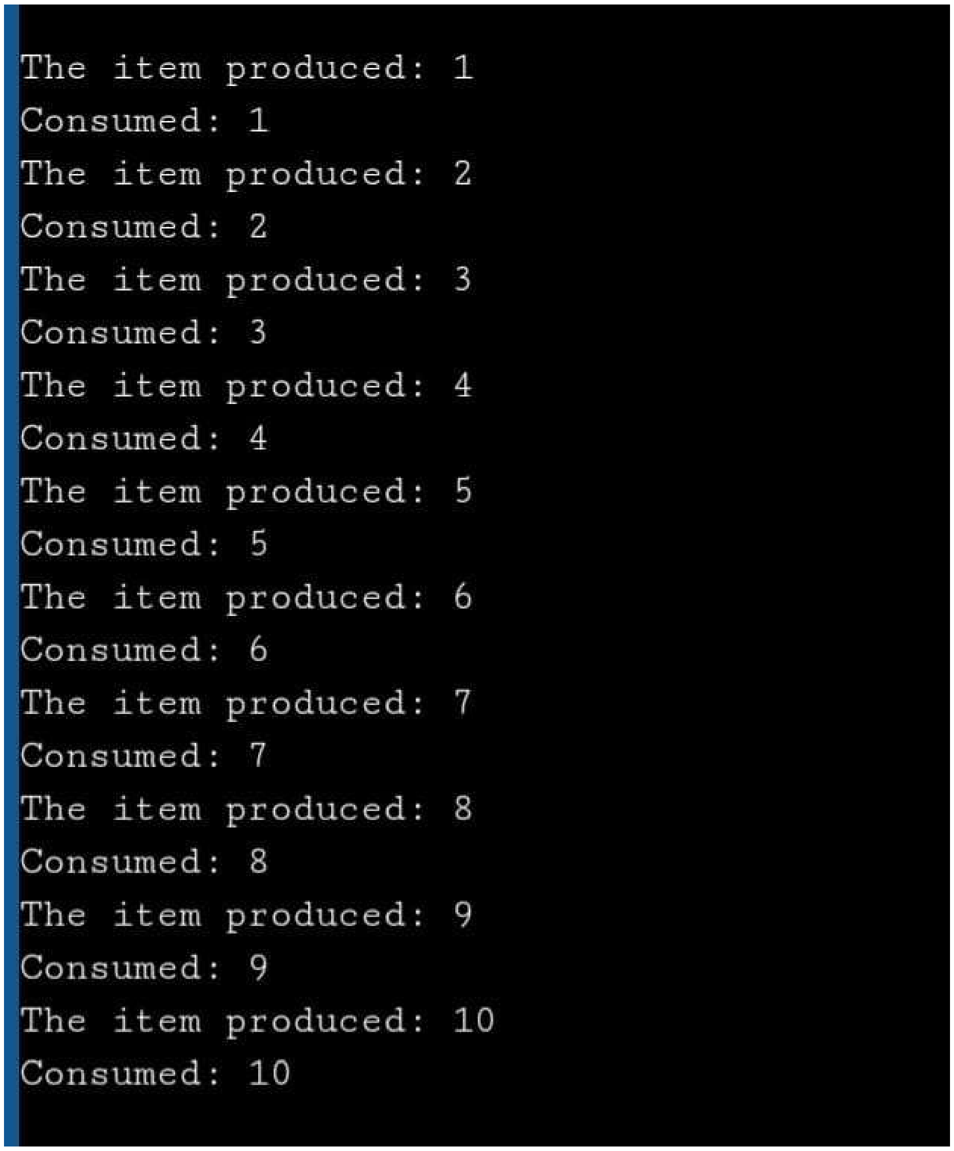
1. Initialize semaphore empty, full and mutex.
2. Create two threads- producer thread and consumer thread.
3. Wait for target thread termination.
4. Call sem_wait on empty semaphore followed by mutex semaphore before entry into critical section.
5. Produce/Consume the item in critical section.
6. Call sem_post on mutex semaphore followed by full semaphore
7. before exiting critical section.
8. Allow the other thread to enter its critical section.
9. Terminate after looping ten times in producer and consumer Threads each.

Program Code:

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

#define SIZE 5

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

void* producer(void* arg){
   for(int i=0;i<10;i++){
         //entry section
         sem_wait(&empty);
         sem_wait(&mutex);
         //critical section
         buffer[in] = i+1;
         printf("\nThe item produced: %d",i+1);
         in = (in+1) % SIZE;
          //exit section
         sem_post(&mutex);
         sem_post(&full);

         sleep(1);
   }
   pthread_exit(NULL);
}
```

```
void* consumer(void* arg){
    for(int i=0;i<10;i++){
        //entry section
        sem_wait(&full);
        sem_wait(&mutex);
        //entry section
        int item = buffer[out];
        printf("\nConsumed: %d",item);
        out = (out+1)% SIZE;
        // exit section
        sem_post(&empty);
        sem_post(&mutex);

         sleep(1);
    }
    pthread_exit(NULL);
}

int main(){

    pthread_t prod,cons;

    sem_init(&empty, 0,SIZE);
    sem_init(&full,0,0);
    sem_init(&mutex,0,1);


    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);
    sem_destroy(&mutex);

    return 0;

}
```

Output:



```
The item produced: 1
Consumed: 1
The item produced: 2
Consumed: 2
The item produced: 3
Consumed: 3
The item produced: 4
Consumed: 4
The item produced: 5
Consumed: 5
The item produced: 6
Consumed: 6
The item produced: 7
Consumed: 7
The item produced: 8
Consumed: 8
The item produced: 9
Consumed: 9
The item produced: 10
Consumed: 10
```

Result: Thus, the program was executed successfully.