TE IT

Name : Omkar Gurav

Roll no : 8048

Thread synchronization using counting semaphores. Application to demonstrate: producer-consumer problem with counting semaphores and mutex.

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

sem_t empty, full;
pthread_mutex_t pmt;
int arr[5],in=0,out=0;

void* prodFunc(void* arg)
{
 int item=1;
 for(int i=0;i<5;i++)
 {
  sem_wait(&empty);
  pthread_mutex_lock(&pmt);

  printf("Item number : %d\n",(i+1));
  printf("  Producer produced item %d\n", item);
  arr[in]=item;
```

```
  in=(in+1)%5;
  item++;

  pthread_mutex_unlock(&pmt);
  sem_post(&full);
  sleep(1);
 }
}

void* consFunc(void* arg)
{
 int item;
 for(int i=0; i<5; i++)
 {
  sem_wait(&full);
  pthread_mutex_lock(&pmt);

  item=arr[out];
  out=(out+1)%5;
  printf("  Consumer consumed item %d\n\n", item);

  pthread_mutex_unlock(&pmt);
  sem_post(&empty);
  sleep(1);
 }
}

int main()
{
 pthread_t pt1, pt2;
```

```
pthread_mutex_init(&pmt, NULL);
sem_init(&empty, 0, 5);
sem_init(&full, 0, 0);

pthread_create(&pt1, NULL, consFunc, NULL);
pthread_create(&pt2, NULL, prodFunc, NULL);
pthread_join(pt1, NULL);
pthread_join(pt2, NULL);

 printf("\nCompleted\n");

 return 0;
}
```

Output:

```
Item number : 1
  Producer produced item 1
  Consumer consumed item 1

Item number : 2
  Producer produced item 2
  Consumer consumed item 2

Item number : 3
  Producer produced item 3
  Consumer consumed item 3
```

Item number : 4

Producer produced item 4

Consumer consumed item 4

Item number : 5

Producer produced item 5

Consumer consumed item 5

Completed