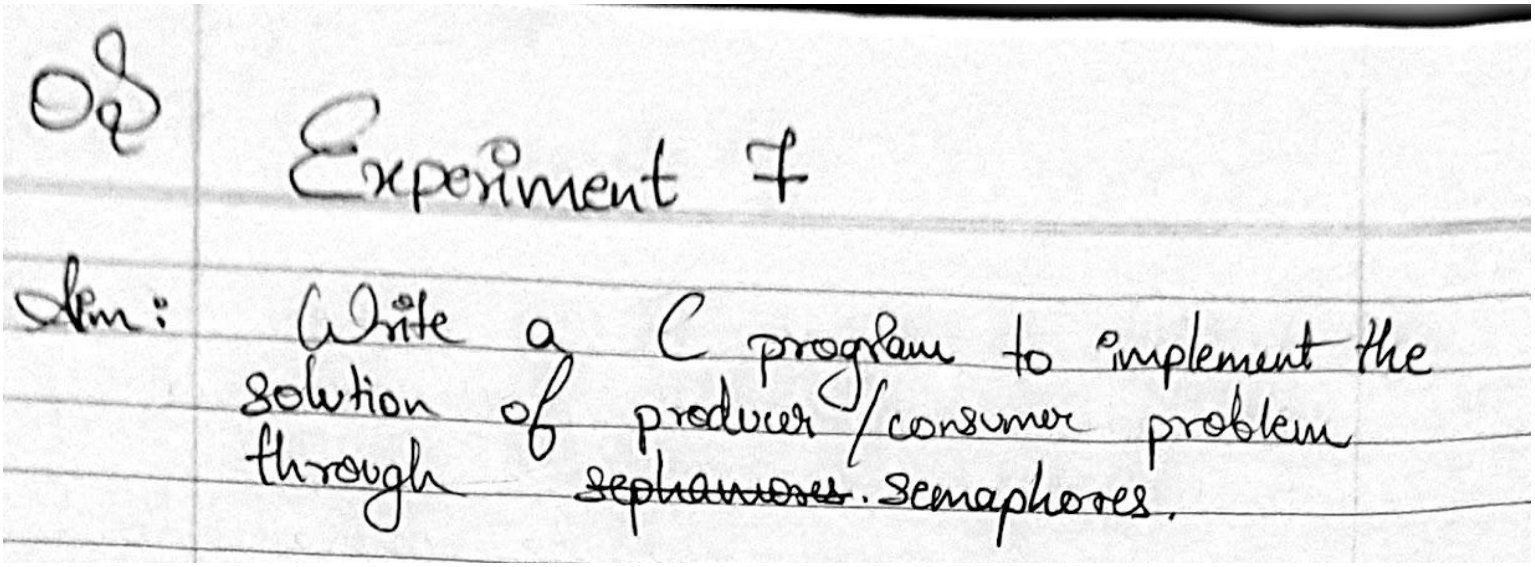
Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department.
OS / Even Sem 2021-22 / Experiment 7.

YASH SARANG.
47 / D6AD.
EXPERIMENT - 7.
Semaphores.

Aim: Write a C program to implement the solution of Producer consumer problem through Semaphore.

Theory:


OS
Experiment 7
Aim: Write a C program to implement the solution of producer/consumer problem through ~~sephamores~~. semaphores.

Theory: Sephamore is an OS and programming language software approach or mechanism, that are used to provide concurrency.

Semaphores was proposed by Djikstra in 1965 which is a very significant technique to manage concurrency by using a simple integer value, which is known as a semaphore.

It is the integer variable that is shared between threads. This variable is used to solve the critical section problem and to achieve process synchronization in multiprocessing environment.

They are of 2 types:

i) Binary: Also known as muter lock, can only have two values - 0 & 1. Its value is initialised to 1. used to implement the solution of critical section problems with multiple processes.

ii) Counting: Its value can range over an unrestricted domain. It is used to control access to a service that has multiple instances.

An inadequate solution can result in a deadlock where both processes are waiting to be awakened.

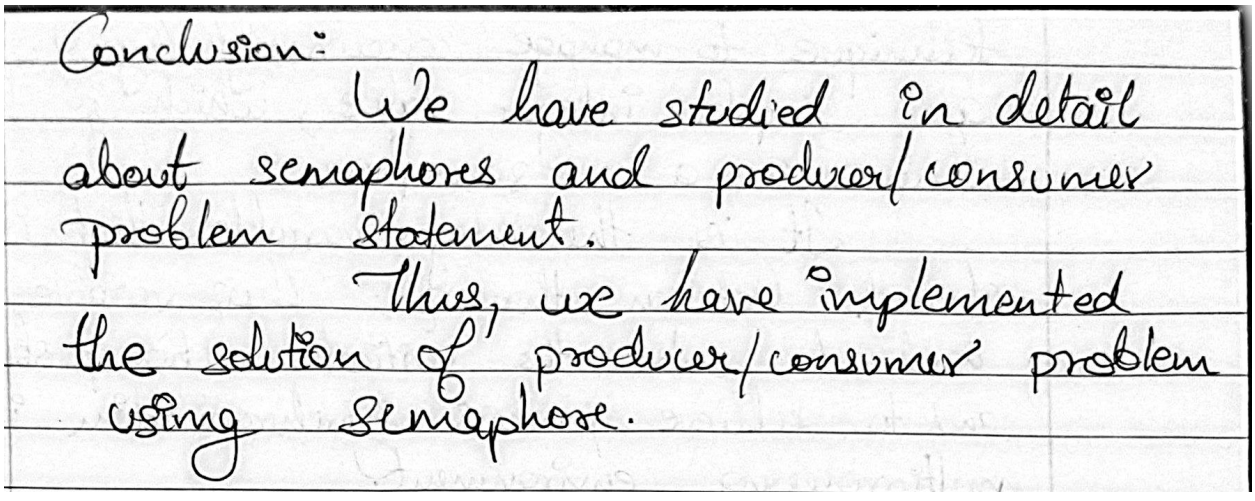
Problems can also be generated to have multiple producers and consumers in a program.

Program Code:

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

#define MaxItems 5 // Maximum items a producer can produce or a consumer can consume
#define BufferSize 5 // Size of the buffer

sem_t empty;
sem_t full;
int in = 0;
int out = 0;
int buffer[BufferSize];
pthread_mutex_t mutex;

void *producer(void *pno)
{
    int item;
    for(int i = 0; i < MaxItems; i++) {
        item = rand(); // Produce an random item
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = item;
        printf("Producer %d: Insert Item %d at %d\n", *((int *)pno),buffer[in],in);
        in = (in+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
}
void *consumer(void *cno)
{
    for(int i = 0; i < MaxItems; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
        printf("Consumer %d: Remove Item %d from %d\n",*((int *)cno),item, out);
        out = (out+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}
```

```
        printf("Consumer %d: Remove Item %d from %d\n",*((int *)cno),item, out);
        out = (out+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}

int main()
{

    pthread_t pro[5],con[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty,0,BufferSize);
    sem_init(&full,0,0);

    int a[5] = {1,2,3,4,5}; //Just used for numbering the producer and consumer

    for(int i = 0; i < 5; i++) {
        pthread_create(&pro[i], NULL, (void *)producer, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&con[i], NULL, (void *)consumer, (void *)&a[i]);
    }

    for(int i = 0; i < 5; i++) {
        pthread_join(pro[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(con[i], NULL);
    }

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

    return 0;

}
```

Conclusion:-



Output:-

```
Producer 1: Insert Item 1804289383 at 0
Producer 2: Insert Item 1681692777 at 1
Consumer 3: Remove Item 1804289383 from 0
Consumer 3: Remove Item 1681692777 from 1
Producer 2: Insert Item 424238335 at 2
Producer 2: Insert Item 1649760492 at 3
Producer 4: Insert Item 846930886 at 4
Consumer 1: Remove Item 424238335 from 2
Consumer 1: Remove Item 1649760492 from 3
Consumer 1: Remove Item 846930886 from 4
Producer 5: Insert Item 1957747793 at 0
Producer 5: Insert Item 1025202362 at 1
Producer 5: Insert Item 1350490027 at 2
Producer 5: Insert Item 783368690 at 3
Consumer 2: Remove Item 1957747793 from 0
Consumer 2: Remove Item 1025202362 from 1
Consumer 2: Remove Item 1350490027 from 2
Consumer 2: Remove Item 783368690 from 3
Producer 5: Insert Item 1102520059 at 4
Consumer 1: Remove Item 1102520059 from 4
Producer 3: Insert Item 1714636915 at 0
Producer 3: Insert Item 2044897763 at 1
Producer 3: Insert Item 1967513926 at 2
Producer 3: Insert Item 1365180540 at 3
Producer 3: Insert Item 1540383426 at 4
Consumer 2: Remove Item 1714636915 from 0
Consumer 1: Remove Item 2044897763 from 1
Consumer 4: Remove Item 1967513926 from 2
Consumer 4: Remove Item 1365180540 from 3
Consumer 4: Remove Item 1540383426 from 4
Producer 2: Insert Item 596516649 at 0
Producer 2: Insert Item 304089172 at 1
Consumer 4: Remove Item 596516649 from 0
Consumer 4: Remove Item 304089172 from 1
Producer 1: Insert Item 719885386 at 2
Producer 1: Insert Item 1303455736 at 3
Producer 1: Insert Item 35005211 at 4
Producer 1: Insert Item 521595368 at 0
Producer 4: Insert Item 1189641421 at 1
Consumer 3: Remove Item 719885386 from 2
Consumer 3: Remove Item 1303455736 from 3
Consumer 3: Remove Item 35005211 from 4
Consumer 5: Remove Item 521595368 from 0
```