OPERATING SYSTEM LABORATORY MANUAL

UNIVERSITY OF THE PUNJAB

FACULTY OF COMPUTING & INFORMATION TECHNOLOGY, LAHORE

DEPARTMENT OF COMPUTER SCIENCE

Course:	**Operating System Lab**	**Date:**
Course Code:	**CC-217-3L**	**Max Marks: 40**
Faculty/Instructor's Name & Email:	**Dr. Ahmad Hassan Butt (ahmad.hassan@pucit.edu.pk)**	

LAB MANUAL # 12

(SPRING 2023)

Name:_____ Enroll No: _____

Objective(s) :

To implement Producer & Consumer Problem (Semaphore).

Lab Tasks :

Task 01: Write the code to initialize Semaphore Mutex.

Task 02: Write the code according the requirement if it's a consumer.

Task 03: Write the code according to the requirement if it's a producer.

Task 04: Display the result.

Lab Grading Sheet :

Task	Max Marks	Obtained Marks	Comments(*if any*)
1.	10		
2.	10		
3.	10		
4.	10		
Total	**40**		**Signature**

Note : Attempt all tasks and get them checked by your Instructor

Lab 12: Semaphore

Objective(s):

To implement Producer & Consumer Problem (Semaphore)

Tool(s) used:

Ubuntu, VIM Editor

A **semaphore**, in its most basic form, is a protected integer variable that can facilitate and restrict access to shared resources in a multi-processing environment. The producer–consumer problem is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer used as a queue. The producer's job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

Task 01: Write the code for the Semaphore mutex, full & empty initialized.

Task 02: Write the code in the case of producer process.

i) Produce an item in to temporary variable.

ii) If there is empty space in the buffer check the mutex value for enter into the critical section.

iii) If the mutex value is 0, allow the producer to add value in the temporary variable to the buffer.

Task 03: Write the code in the case of consumer process.

i) It should wait if the buffer is empty

ii) If there is any item in the buffer check for mutex value, if the mutex==0, remove item from buffer

iii) Signal the mutex value and reduce the empty value by 1.

iv) Consume the item.

Task 04: Print the result.

Program

```
#define BUFFERSIZE 10
int
mutex,n,empty,full=0,item,item
1;
int buffer[20];
int
in=0,out=0,mutex=1;
void wait(int s)
{
  while(s<0)
  {
     printf("\nCannot add an
     item\n"); exit(0);
  }
  s--;
}
void signal(int s)
{
  s++;
}
void producer( )
{
  Do
  {
   wait (empty);
   wait(mutex);
   printf("\nEnter an
   item:");
```

```
   scanf("%d",&item);
   buffer[in]=item;
   in=in+1;
   signal(mutex);
   signal(full);
  }
while(in<n);
}

void consumer( )
{
 Do
{
  wait(full);
  wait(mutex);
  item1=buffer[out]; printf("\nConsumed item
=%d",item1);
  out=out+1;
  signal(mutex);
  signal(empty);
 }
 while(out<n);
}
 void main( )
 {
      printf("Enter the value of n:");
      scanf("%d ",&n);
      empty=n;
      while(in<n)
           producer( );
```

```
    while(in!=out)
        consumer();
}
```

OUTPUT