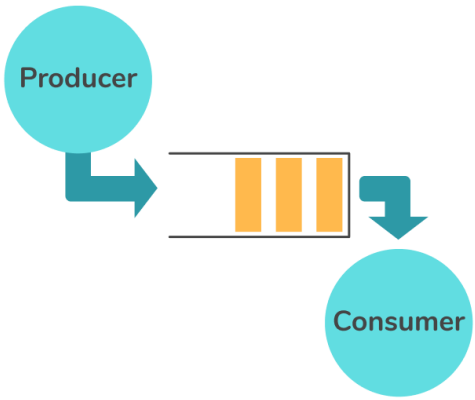
Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Operating System Laboratory
Course Code:	DJ19CEL403
Experiment No.:	05

AIM: **PRODUCER CONSUMER PROBLEM (SEMAPHORE).**

THEORY:

What is Producer Consumer Problem?



- In operating System **Producer is a process which is able to produce** data/item.
- **Consumer is a Process that is able to consume** the data/item produced by the Producer.
- Both Producer and Consumer **share a common memory buffer**. This buffer is a space of a certain size in the memory of the system which is used for storage. The producer produces the data into the buffer and the consumer consumes the data from the buffer.

- So, what are the Producer-Consumer Problems?
  - Producer Process should not produce any data when the shared buffer is full.
  - Consumer Process should not consume any data when the shared buffer is empty.
  - The access to the shared buffer should be mutually exclusive i.e at a time only one process should be able to access the shared buffer and make changes to it.
- For consistent data synchronization between Producer and Consumer, the above problem should be resolved.

- Solution For Producer Consumer Problem
  To solve the Producer-Consumer problem three semaphores variable are used :
  - Semaphores are variables used to indicate the number of resources available in the system at a particular time
  - Semaphore variables are used to achieve `Process Synchronization.
- Full
  - The full variable is used to track the space filled in the buffer by the Producer process. It is initialized to 0 initially as initially no space is filled by the Producer process.
- Empty
  - The Empty variable is used to track the empty space in the buffer. The Empty variable is initially initialized to the BUFFER-SIZE as initially, the whole buffer is empty.

- Mutex
  - Mutex is used to achieve mutual exclusion. mutex ensures that at any particular time only the producer or the consumer is accessing the buffer.
- Mutex - mutex is a binary semaphore variable that has a value of 0 or 1.
- The Signal() and wait() operation in the above-mentioned semaphores is used to arrive at a solution to the Producer-Consumer problem.
- Signal()
  - The signal function increases the semaphore value by 1. Wait() - The wait operation decreases the semaphore value by 1.
- **As mutexes have binary values i.e 0 and 1. So whenever any process tries to enter the critical section code it first checks for the mutex value by using the wait operation.**

CODE:

```c
#include <stdio.h>
#include <stdlib.h>

int mutex = 1;
int products = 0;
int emptyspace = 5, x = 0;

int Wait(int n)
{
    --n;
    return n;
}

int Signal(int n)
{
    ++n;
    return n;
}

void producer()
{
    mutex = Wait(mutex);
    products = Signal(products);

    emptyspace = Wait(emptyspace);
    x = Signal(x);

    printf("Producer produces item : %d\n", x);
    mutex = Signal(mutex);
}
```

```
void consumer()
{
    mutex = Wait(mutex);
    products = Wait(products);
    emptyspace = Signal(emptyspace);
    printf("Consumer consumes item : %d\n", x);
    x = Wait(x);
    mutex = Signal(mutex);
}

int main()
{
    int n, i = 1;
    printf("\n1. Press 1 for Producer\n2. Press 2 for Consumer\n3. Press 3 for
Exit");
    while (i > 0)
    {
        printf("\nEnter your choice: ");
        scanf("%d", &n);

        switch (n)
        {

        case 1:
            if ((mutex == 1) && (emptyspace != 0))
                producer();
            else
                printf("Buffer is Full\nProducer cannot produce more
items\n");
            break;

        case 2:
            if ((mutex == 1) && (products != 0))
                consumer();
            else
                printf("Buffer is empty\nConsumer not allowed to consume\n");
            break;

        case 3:
            exit(0);
            break;
        }
    }
}
```

OUTPUT:

```
1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice: 2
Buffer is empty
Consumer not allowed to consume

Enter your choice: 1
Producer produces item : 1

Enter your choice: 1
Producer produces item : 2

Enter your choice: 1
Producer produces item : 3

Enter your choice: 1
Producer produces item : 4

Enter your choice: 1
Producer produces item : 5

Enter your choice: 1
Buffer is Full
Producer cannot produce more items

Enter your choice: 2
Consumer consumes item : 5
```

```
Enter your choice: 2
Consumer consumes item : 4

Enter your choice: 2
Consumer consumes item : 3

Enter your choice: 2
Consumer consumes item : 2

Enter your choice: 2
Consumer consumes item : 1

Enter your choice: 2
Buffer is empty
Consumer not allowed to consume

Enter your choice: 1
Producer produces item : 1

Enter your choice: 2
Consumer consumes item : 1

Enter your choice: 2
Buffer is empty
Consumer not allowed to consume
```

CONCLUSION:

- Producer Process produces data item and consumer process consumes data item.
- Both producer and consumer processes share a common memory buffer.
- Producer should not produce any item if the buffer is full.
- Consumer should not consume any item if the buffer is empty.
- Not more than one process should access the buffer at a time i.e. mutual exclusion should be there.
- Full, Empty and mutex semaphore help to solve Producer-consumer problem.
- Full semaphore checks for the number of filled space in the buffer by the producer process
- Empty semaphore checks for the number of empty spaces in the buffer.
- Mutex checks for the mutual exclusion.