## Experiment 06- To write a program to implement the solution of process synchronization using Semaphore

**Learning Objective:** Students should be able to understand process synchronization by using different coding languages, such as C/C++/Java/Python.

**Tools:** Online compiler

**Theory:** Process synchronization ensures that multiple processes execute in a controlled manner when sharing resources. Semaphores are synchronization tools used to manage concurrent processes by signaling when a resource is available or occupied. They use two operations: P (wait) and V (signal) to coordinate process execution and prevent race conditions.

A **semaphore** is a synchronization tool that consists of:

1. An integer value that indicates available resources.
2. A waiting queue for processes that need to access a resource.
3. Two main operations:
    - **P (Wait/Down operation):** If the semaphore value is greater than zero, the process can proceed; otherwise, it gets blocked.
    - **V (Signal/Up operation):** If a process is waiting in the queue, it is woken up; otherwise, the semaphore value is incremented.

**Code:**

```
#include <stdio.h>
#include<stdlib.h>
#include<sys/queue.h>
struct semaphore{
Queue<process> q;
int value;
};

void P(struct semaphore s)
{
   if (s.value == 1) {
s.value = 0;     }
else {
       s.q.push(P);
sleep();
   }
}

void V(semaphore s)
{
```

```
   if (s.q is empty) {
s.value = 1;     }
else {

       // Get a process from the Waiting Queue
       Process p = q.front();
       // Remove the process from waiting
q.pop();         wakeup(p);
   }}
int main() {     printf("This is
Neev!!");
 return 0;
}
```

**Output Screenshot:** This is Neev!!

**Learning Outcomes:** The student should have the ability to:

LO2.1 Outline various compilers for different language
LO2.2 Understood the process synchronization
LO2.3 Choose an appropriate compiler to solve the process synchronization problem

**Course Outcomes:** Upon completion of the course students will be able to learn about operating systems and security concepts.

**Conclusion:** The experiment successfully demonstrates process synchronization using semaphores. By implementing the **P()** and **V()** operations, we ensure that processes do not interfere with each other while accessing shared resources. This helps maintain data consistency and avoids deadlocks.

For Faculty Use:

| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance/ Learning Attitude [20%]** | |
|---|---|---|---|---|
| **Marks Obtained** | | | | |