**EXPERIMENT-5**

**Aim: -** To study implement Peterson’s Algorithm

**Learning Objective:** Understand the concept of Peterson’s Algorithm

**Theory: -**

The producer consumer problem (or bounded buffer problem) describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue. Producer produce an item and put it into buffer. If buffer is already full then producer will have to wait for an empty block in buffer. Consumer consume an item from buffer. If buffer is already empty then consumer will have to wait for an item in buffer. Implement Peterson’s Algorithm for the two processes using shared memory such that there is mutual exclusion between them. The solution should have free from synchronization problems.

**Explanation of Peterson’s algorithm –**

Peterson’s Algorithm is used to synchronize two processes. It uses two variables, a bool array **flag** of size 2 and an int variable **turn** to accomplish it.  
In the solution i represents the Consumer and j represents the Producer. Initially the flags are false. When a process wants to execute it’s critical section, it sets it’s flag to true and turn as the index of the other process. This means that the process wants to execute but it will allow the other process to run first. The process performs busy waiting until the other process has finished it’s own critical section.  
After this the current process enters it’s critical section and adds or removes a random number from the shared buffer. After completing the critical section, it sets it’s own flag to false, indication it does not wish to execute anymore.

The program runs for a fixed amount of time before exiting. This time can be changed by changing value of the macro RT.

**Code:**

import threading

N = 2

flag = [False] \* N

turn = 0

def producer(j):

    while True:

        flag[j] = True

        turn = 1 - j

        while flag[1 - j] and turn == 1 - j:

            pass

        flag[j] = False

def consumer(i):

    while True:

        flag[i] = True

        turn = i

        while flag[1 - i] and turn == i:

            pass

        flag[i] = False

producer\_thread = threading.Thread(target=producer, args=(0,))

consumer\_thread = threading.Thread(target=consumer, args=(1,))

producer\_thread.start()

consumer\_thread.start()

producer\_thread.join()

consumer\_thread.join()

**Lab Outcome**: Students are able to understand Peterson’s algorithm.

**Conclusion**: Peterson’s Algorithm for the two processes using shared memory such that there is mutual exclusion between them. The solution should have free from synchronization problems.