27. Write a program for semaphore signaling mechanism where a process can signal a process that is waiting on a semaphore.

Test Case:

number of instances: 4

Number of Processes: 4 (P1, P2, P3, P4) all are calling wait operation on S

Show the response when another process P5 wants the same resource.

Program:-

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

sem\_t semaphore;

// Function executed by each process

void\* processFunction(void\* arg) {

int processID = \*(int\*)arg;

printf("Process P%d is waiting...\n", processID);

sem\_wait(&semaphore); // Wait operation on semaphore

// Critical section

printf("Process P%d entered the critical section.\n", processID);

printf("Process P%d is using the shared resource.\n", processID);

sem\_post(&semaphore); // Signal operation on semaphore

printf("Process P%d exited the critical section.\n", processID);

pthread\_exit(NULL);

}

int main() {

int numInstances = 4;

int numProcesses = 4;

sem\_init(&semaphore, 0, numInstances);

pthread\_t threads[numProcesses];

int processIDs[] = {1, 2, 3, 4};

// Create threads for each process

for (int i = 0; i < numProcesses; i++) {

pthread\_create(&threads[i], NULL, processFunction, (void\*)&processIDs[i]);

}

// Wait for all threads to finish

for (int i = 0; i < numProcesses; i++) {

pthread\_join(threads[i], NULL);

}

// Process P5 wants the same resource

int processID = 5;

printf("Process P%d is waiting...\n", processID);

sem\_wait(&semaphore); // Wait operation on semaphore

// Critical section

printf("Process P%d entered the critical section.\n", processID);

printf("Process P%d is using the shared resource.\n", processID);

sem\_post(&semaphore); // Signal operation on semaphore

printf("Process P%d exited the critical section.\n", processID);

sem\_destroy(&semaphore);

return 0;

}

Output:-

