Program 3:

```
/* Program for process synchronization using semaphores
Program create two threads: one to increment the value of a shared variable and second to
decrement the value of shared variable. Both the threads make use of semaphore variable so that
only one of the threads is executing in its critical section */

#include<pthread.h>
#include<stdio.h>
#include<semaphore.h>
void *fun1();
void *fun2();
int shared=1; //shared variable
sem_t s; //semaphore variable
int main()
{
sem_init(&s,0,1); //initialize semaphore variable - 1st argument is address of variable, 2nd is
number of
                        //processes sharing semaphore, 3rd argument is the initial value of semaphore
variable
pthread_t thread1, thread2;
pthread_create(&thread1, NULL, fun1, NULL);
pthread_create(&thread2, NULL, fun2, NULL);
pthread_join(thread1, NULL);
pthread_join(thread2,NULL);
printf("Final value of shared is %d\n",shared); //prints the last updated value of shared variable
}

void *fun1()
{
    int x;
    sem_wait(&s); //executes wait operation on s
    x=shared;//thread one reads value of shared variable
    x++;  //thread one increments its value
    sleep(1);  //thread one is preempted by thread 2
    shared=x; //thread one updates the value of shared variable
    sem_post(&s); //executes signal operation on s
}

void *fun2()
{
    int y;
    sem_wait(&s);
    y=shared;
    y--;
    sleep(1);
    shared=y;
    sem_post(&s);
}

/* the final value of shared variable will be 1. When any one of the threads execute the wait
operation the value of "s" becomes zero and hence the other thread (even if it preempts the
```

running thread) is not able to successfully execute the wait operation on "s" thus not able to read the inconsistent value of shared variable. Thus only one of the thread is running in its critical section at any given time */