# Worksheet 6

**1/22/2025**

15 Points Possible

Attempt 1

1/22/2025

**NEXT UP: Review Feedback**

Attempt 1 Score: N/A

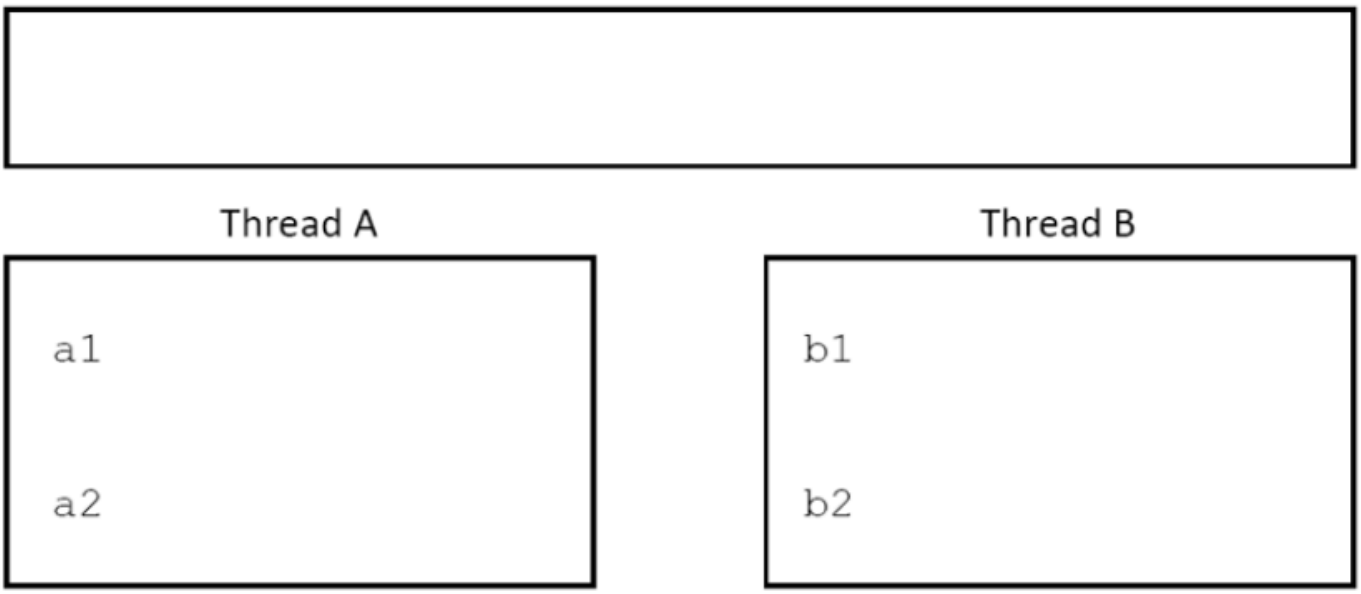
Add Comment

**Unlimited Attempts Allowed**

## Details

Q1: Use one or more semaphores to achieve the following order of execution:

- a1 must execute before b2
- b1 must execute before a2




**Q2:** Fill up the blanks in the Semaphore Up/Down methods, which are equivalent to increment() and decrement(). Use the following lines to fill up the code (match each code with one or more blank).

1. AddToEnd (t)
2. oldIntStat = SetInterruptsTo (DISABLED)
3. waitingThreads.AddToEnd (currentThread)
4. t = waitingThreads.Remove ()
5. Sleep ()

```
----------  Semaphore.Up  ----------
method Up ()
    var
      oldIntStat: int
      t: ptr to Thread
     A.
    _____________________________

    if count == 0x7fffffff
      FatalError ("Semaphore count overflowed during 'Up' operation")
    endIf
    count = count + 1
    if count <= 0
     B.
    _____________________________

      t.status = READY
      readyList.AddToEnd (t)
    endIf
    oldIntStat = SetInterruptsTo (oldIntStat)
  endMethod
```

**Q3: Fill up the blanks in the Semaphore Up/Down methods, which are equivalent to increment() and decrement(). Use the following lines to fill up the code (match each code with one or more blank).**

1. readyList.AddToEnd (t)
2. oldIntStat = SetInterruptsTo (DISABLED)
3. waitingThreads.AddToEnd (currentThread)
4. t = waitingThreads.Remove ()
5. Sleep ()

```
----------  Semaphore . Down  ----------
method Down ()
    var
      oldIntStat: int

     A.
    ____________________________

    if count == 0x80000000
      FatalError ("Semaphore count underflowed during 'Down' operation")
    endIf
    count = count - 1
    if count < 0

     B.
    ____________________________

     C.
    ____________________________

     endIf
    oldIntStat = SetInterruptsTo (oldIntStat)
  endMethod
```

**Answer 1 (Q1):**

We will use two semaphore variables (sem1, sem2) initialized to "0" to achieve the two execution orders.

- sem1: to achieve "a1 must execute before b2"
- sem2: to achieve "b1 must execute before a2"

| |
|---|
| **sem1 = Semaphore(0)**<br>**sem2 = Semaphore(0)** |

To make sure a1 is completed before b2, we will decrement sem1 before b2 so that sem1 becomes negative and does not execute b2 and increment sem1 before a1 so that it can execute a1 without issues.

To make sure b1 is completed before a2, we will decrement sem2 before a2 so that sem2 becomes negative and does not execute a2 and increment sem2 before b1 so that it can execute b1 without issues.

| Thread A | Thread B |
|---|---|
| a1<br>**sem1.increment()**<br>**sem2.decrement()**<br>a2 | b1<br>**sem2.increment()**<br>**sem1.decrement()**<br>b2 |

**Answer 2 (Q2):**

**A. oldIntStat = SetInterruptsTo (DISABLED)**
**B. t = waitingThreads.Remove ()**

- The first thing we do in semaphore is, disable the interrupt. So, we need, **oldIntStat = SetInterruptsTo (DISABLED) in A**.
- If count <= 0: we check if the list is not empty, so we wake up it after this check. To wake it up, we will remove it from the waiting threads list. So, **t = waitingThreads.Remove () in B**.

**Answer 3 (Q3):**

**A. oldIntStat = SetInterruptsTo (DISABLED)**
**B. waitingThreads.AddToEnd (currentThread)**
**C. Sleep ()**

- The first thing we do in semaphore is, disable the interrupt. So, we need, **oldIntStat = SetInterruptsTo (DISABLED) in A**.
- If count is negative (count < 0), we go to the waiting list of threads and add it so that it can get woken up later. We need to add the thread to this waiting list before we put it to sleep, so, **waitingThreads.AddToEnd (currentThread) in B.**
- Then we put it to sleep. So, **Sleep () in C.**

New Attempt