Name: Bing ying Liang

ID: 48999397

CS 7343

Distance Student.

```
    semaphore n=0, s=1;
1.  void producer()
    {
        while (true)
        {
            producer();
            semWait(s);
            append();
            semSignal(s);
            semSignal(n);
        }
    }

    void consumer()
    {
        while (true)
        {
            semWait(n);
            semWait(s);
            take();
            semSignal(s);
            consume();
        }
    }
```

2. a) Sol:

Ready queue [A] [A] [AB] [BA] [CB] [BC] [CDB] [BCD] [DEBC] [CDEB] [BCDE] ....

CPU [A] [A] [B] [A] [B] [C] [B] [D] [C] [B] [E] ....

RR q=1

Therefore. turn around time for A is 4

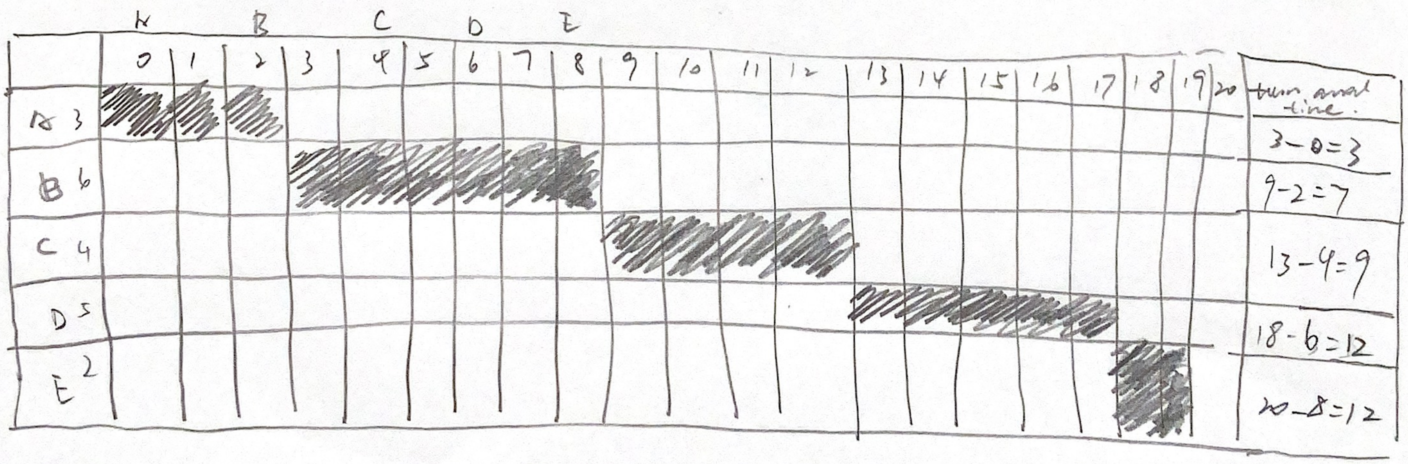
B is 16

C is 13

D is 14

E is 7

b)



FCFS

Therefore, turn around time for A is 3

B is 7

C is 9

D is 12

E is 12

c)

SPN

Therefore, turn around time time for

A is 3

B is 7

C is 9

D is 14

E is 3

3. sol

	FCFS	ROUND Robin	SPN/SJF	SRT	HRRN
Selection Function	max[w]	constant	min[s]	min[s-e]	$\max\left(\frac{w+s}{s}\right)$
Decision Mode	Non-preemptive	Preemptive (at time quantum)	Non-preemptive	Preemptive (at arrival)	Non-preemptive
Response Time	May be high, especially if there is a large variance in process execution times	Provides good response time for short processes	Provides good response time for short processes	Provides good response time	Not emphasized
Starvation	No	No	Possible	possible	NO

4. sol : a)

5. (1) Two Threads (call them T1, T2)

void main() { tally = 0; parbegin (total(), total()); write (tally);

(a) For upper bound: 100.

Explain: both T1, T2 run sequentially. in this case, T1 will run into completion followed by T2 as in.

(2) For lower bound: 2

Explain: In assembly language we will have
① lw R0, tally
② add R0, 1
③ sw R0, tally

T1

n=1
① lw R0, tally=0
② add R0, 1
③ sw R0, tally=1

T2

n=1
① lw R0, tally=0
② add R0, 1 - - - - - - switch to T1
③ sw R0, tally = 1

⋮

n=49 finish

n=50
① lw R0, tally=1
② add R0, 1 - - - - switch to T2 →

T2 finish Tally=50

③ sw R0, tally = $\boxed{2}$

When T_2 finish ① lw ② add then switch to T_1, T_2 has not write R_0 to tally,
Now T_1 keep run until $n=49$. it still has last turn. For now global tally = 49
But this time switch T_2 to run ③ write $R_0 = 1$ to tally. and Now global tally = 1.
Switch to T_1 to finish last turn. T_1 finish ① ② and then switch to T_2 to finishes. T_2 finishes, the global tally = 50. Now T_1 start to finish ③ sw $R_0=2$ to tally. The result is tally = 2.
Therefore, lower bound is 2, upper bound is 100

---

(2) Three Threads

Call them T_1, T_2, T_3

(a) upper bound is 150
Explain: T_1, T_2, T_3 run sequentially. T_1 finishes. Tally = 50, Then T_2 use Tally = 50 to keep finishing its thread, the result will be tally = 100. Then T_3 use Tally = 100 as start to finishes, the result will be tally = 150.

(b) low bound is 2
Explain: In assembly language we will have
① lw R_0, tally
② add R_0, 1
③ sw R_0, tally.

T_1	T_2	T_3	
① lw R_0, tally = 0	① lw R_0, tally = 0	① lw R_0, tally = 0	tally = 0
n = 1 ② add R_0, 1	n = 1 ② add R_0, 1	② add R_0, 1	
③ sw R_0, tally = 1			
⋮			
n = 49 finish			tally = 49
	③ sw $R_0 = 1$, tally = 1		tally = 1
n = 50 ① lw, R_0, tally = 1			
② add, $R_0 = 1$, 1 // $R_0 = 2$			
	finishes n = 50 tally = 50		tally = 50
		③ sw $R_0 = 1$, tally = 1	tally = 1
		T_3 finishes tally = 50	
③ sw, $R_0 = 2$, tally = 2			

6. sol: If T1 is running.

It enters wait on the semaphore S. checks which

assume false loop end.

Now T2, same thing. back to T1.

T1 continues decreasing. (no check, because loop is over).

exit and wait.

→ T2

Now they have ~~both~~ both enter critical section.

7. Sol: (a) illegal. The scheduler selects threads to run from the list of ready (or runable) threads. A blocked thread must first be placed in the ready queue before it can be selected to run

(b) Legal. A running thread can become blocked when it requests a resource that is not immediately available (disk I/O, lock, etc).

(c) ILLegal. A thread can only transition to Blocked from Running. It cannot execute any statements when still in a queue: for example, ready queue.