CAS CS 350 HW7

Andrea Lopez

TOTAL POINTS

97 / 100

QUESTION 1

1 Q1 **40 / 40**

✓ - **0 pts** Correct

a)

- **4 pts** Incorrect
- **5 pts** No answer provided

b)

- **4 pts** Incorrect
- **5 pts** No answer provided

c)

- **3 pts** minor error in reasoning/unclear reasoning
- **5 pts** Correct answer but reasoning not provided
- **8 pts** Incorrect
- **10 pts** no answer provided

d)

- **3 pts** Minor errors/unclear reasoning
- **5 pts** Missing sample of execution/missing reasoning
- **8 pts** Incorrect
- **10 pts** No answer provided

e)

- **4 pts** incorrect/Missing semaphore initialization
- **4 pts** Incorrect semaphore usage
- **8 pts** Incorrect
- **10 pts** No answer provided

QUESTION 2

2 Q2 **27 / 30**

- **0 pts** Correct

a)

✓ - **1 pts** **Incorrect/Miss part of semaphore initialization**

- **1 pts** Miss 1 essential semaphore lines
- **2 pts** Miss 2 essential semaphore lines
- **3 pts** Miss 3 essential semaphore lines
- **4 pts** Miss 4 essential semaphore lines
- **5 pts** Miss 5 essential semaphore lines
- **6 pts** No answer provided
- **2 pts** logical error

b)

- **1 pts** Miss part of semaphore initialization
- **1 pts** Miss 1 essential semaphore lines
- **2 pts** Miss 2 essential semaphore lines
- **3 pts** Miss 3 essential semaphore lines
- **4 pts** Miss 4 essential semaphore lines
- **5 pts** Miss 5 essential semaphore lines
- **6 pts** No answer provided
- **2 pts** logical error

c)

✓ - **2 pts** **Missing/incorrect initialization**

- **2 pts** Minor error for the code of roommate
- **4 pts** Incorrect code for the roommate
- **2 pts** Minor error for the code of employee
- **4 pts** Incorrect code for the employee
- **6 pts** No answer for employee
- **18 pts** no answer provided

QUESTION 3

3 Q3 **30 / 30**

✓ - **0 pts** Correct

Problem 1

a) equal them both to false to guarantee 2-party mutual exclusion:

flag [i] = false
flag [j] = false

b) Initializing turn to j will have no impact because j is not trying to acquire the critical section. Therefore, process i still has access to the critical section so no effect is seen.

c) No, the output would not be possible because line 7 is not executed! This is because flag[i] is set to true, but in order for j to go to the critical section, flag[i] must be set to false. However, this doesn't happen because once j is done with the critical section, it sets turn to i, so i would have to go through the critical section to then set turn back to j. However, process i is not in the CS, so the output in line 7 is impossible.

d) Adding a break would cause a problem because it would make the flagging i or j not function properly, making it possible for there to be 2 process in the CS at the same time.

example execution:

Problem 1 continued...

d)

	Process i	Process j
1	flag[i] = true ;	1
2	print_line ("Process i: entering CS");	2
3	while (flag[j])	3
4	↳ //not executed	4
5	print_line("Process i: inside CS");	5
6		6 flag[j] = true
7		7 print_line ("process j: entering CS");
8		8 while (flag[i]) {
9		9 if (turn == i) {
10		10 flag[j] = false ;
11		11 break;}
12		12 }
13		13 pint_line (Process j: inside CS");
14	turn = j	14
15	flag[i] = false;	15
16		16 turn = i;
17		17 flag[j] = false;

* Both processes are inside the CS (so this doesn't work)!!

Problem 1 continued...

e) Modified code:

```
Semaphore sem = 1;

Process i:
  repeat:
     // remainder section
     wait(sem);
     // critical section starts!
     print_line ("Process i: Inside cs");
     // end of critical section!
     signal(sem);
     // remainder section
  forever

Process j:
  repeat:
     // remainder section
     wait(sem)
     // critical section starts!
     print_line (Process j: inside cs");
     // end of critical section!
     signal (sem);
     // remainder section
  forever
```

1 Q1 **40 / 40**

✓ **- 0 pts** Correct

a)

- **- 4 pts** Incorrect
- **- 5 pts** No answer provided

b)

- **- 4 pts** Incorrect
- **- 5 pts** No answer provided

c)

- **- 3 pts** minor error in reasoning/unclear reasoning
- **- 5 pts** Correct answer but reasoning not provided
- **- 8 pts** Incorrect
- **- 10 pts** no answer provided

d)

- **- 3 pts** Minor errors/unclear reasoning
- **- 5 pts** Missing sample of execution/missing reasoning
- **- 8 pts** Incorrect
- **- 10 pts** No answer provided

e)

- **- 4 pts** incorrect/Missing semaphore initialization
- **- 4 pts** Incorrect semaphore usage
- **- 8 pts** Incorrect
- **- 10 pts** No answer provided

Problem 2

```
a) Semaphore pizza = 8;
   int slices = 8;
   Semaphore plates = plates < 8;
  begin:
    wait (pizza);
    wait (plates);
    critical section!
          slices --;
    signal (plate);
  end
b) Semaphore pizza = 8;
   int slices = 8;
   Semaphore plates = plates < 8;
   Semaphore fork = 5;
   Semaphore knife = 3;
  begin:
   wait (pizza);
   wait (plates);
   wait (fork);
   wait (knife);
   critical section!
        slices--;
   signal (plates);
   signal (fork);
   signal (knife);
  end
```

Problem 2 continued...

```
c) semaphore pizza = 8;
   int slices = 8;
   semaphore plates = plates < 8;
   semaphore fork = 5;
   semaphore knife = 3;
   semaphore m = 0;
   semaphore pizzeria = 0;
```

```
"Roomies" process
begin:
   wait(pizza);
   wait(plate);
   wait(fork);
   wait(knife);
   wait(m);
   critical section!
       slices--;
       if (slices <= 0){
           signal(pizzeria);
       }
   signal(pizza);
   signal(knife);
   signal(fork);
   signal(plate);
   signal(m);
end
```

```
Mama Jane process
begin:
   wait(m);
   wait(pizzeria);
   critical section!
       slices = 8;
       for slice in range(8){
           signal(pizza);
       }
   signal(m);
end
```

2 Q2 27 / 30

- **0 pts** Correct

a)

✓ **- 1 pts Incorrect/Miss part of semaphore initialization**

- **1 pts** Miss 1 essential semaphore lines
- **2 pts** Miss 2 essential semaphore lines
- **3 pts** Miss 3 essential semaphore lines
- **4 pts** Miss 4 essential semaphore lines
- **5 pts** Miss 5 essential semaphore lines
- **6 pts** No answer provided
- **2 pts** logical error

b)

- **1 pts** Miss part of semaphore initialization
- **1 pts** Miss 1 essential semaphore lines
- **2 pts** Miss 2 essential semaphore lines
- **3 pts** Miss 3 essential semaphore lines
- **4 pts** Miss 4 essential semaphore lines
- **5 pts** Miss 5 essential semaphore lines
- **6 pts** No answer provided
- **2 pts** logical error

c)

✓ **- 2 pts Missing/incorrect initialization**

- **2 pts** Minor error for the code of roommate
- **4 pts** Incorrect code for the roommate
- **2 pts** Minor error for the code of employee
- **4 pts** Incorrect code for the employee
- **6 pts** No answer for employee
- **18 pts** no answer provided

Problem 3

a) blank 1:

- semaphore c_done = 0;

blank 2:

- remains blank!

blank 3:

- signal(c-done);

blank 4:

- wait(c-done);

blank 5:

- remains blank!

b) blank 1:

- semaphore c_done = 0;

blank 2:

- semaphore mutex = 1;

blank 3:

- remains blank!

blank 4-9:

```
- wait (mutex);
   cnt ++;
   if (cnt == 4) {
        signal(c_done);
    }
    signal(mutex);
```

blank 10:

- wait(c-done);

blank 11:

- remains blank!

Problem 3 continued...

c) blank 1:

- Semaphore C-done[4]= [0,0,0,0];

blank 2:

- Semaphore mutex= 1;

blank 3-4:

- remain blank!

blank 5:

- signal (C_done [i]);

blank 6:

- remains blank!

blank 7:

- wait (C-done [j]);

blank 8:

- wait(mutex);

blank 9:

- signal(mutex);

blank 10:

- remains blank!

3 Q3 30 / 30

✓ - 0 pts Correct