長庚大學108學年度第一學期 作業系統 第三次小考

系級: 姓名: 學號:

1. (30%) In this chapter, we have learned about the problem of "Race Condition." For the following machine code of counter++ and counter--, let the initial value of "counter" be 3. Please show the situation that after run counter++ once and counter-- once, the final result of the "counter" is possible to be 2.

counter++ :	counter--:
r1 = counter	r2 = counter
r1 = r1 + 1	r2 = r2 - 1
counter = r1	counter = r2

Answer:

1. counter++: r1 = counter
2. counter++: r1 = r1 + 1
3. counter--: r2 = counter
4. counter--: r2 = r2 – 1
5. counter++: counter = r1
6. counter--: counter = r2

正確條件：(1) counter++與counter--各自的程式碼必須依序執行，(2)counter++在把r1寫回counter之前，counter--必須先把先把counter值讀入r2，(3) counter++在把r1寫回counter之後，counter--才把r2寫回counter。

2. (30%) There are three processes:
   - P_1: a * b → a
   - P_2: b + c → c
   - P_3: c + d → d

   The access to valuables "b" and "c" must be protected in critical sessions. We now have two semaphores, and they are initialized as S_1=1 and S_2=1. The code of P_1 is provided as follows:

```
wati(S1);
a =  a * b;
signal(S1);
```

   Please provide the code of P_2 and P_3.

Answer:

P_2：

```
wati(S1);
wati(S2);
c =  b + c;
signal(S2);
signal(S1);
```

P_3：

```
wati(S2);
d = c + d;
signal(S2);
```

3. (40%) For the bounded-buffer problem with consumers and producers, the code of consumers is provided as follows. Please provide the code of producer.

Consumer:

```
do {
        wait(full);    /* control buffer availability */
        wait(mutex);   /* mutual exclusion */
        remove an item from buffer to nextp;
        signal(mutex);
        signal(empty);   /* increase item counts */
        consume nextp;
} while (1);
```

Producer:

```
do {
        produce an item in nextp;
        Code Line 1;
        Code Line 2;
        add nextp to buffer;
        Code Line 3;
        Code Line 4;
} while (1);
```

Answer: 看課本或投影片