Bài tập đồng bộ

1. Một biến X được chia sẻ bởi hai tiến trình cùng thực hiện đoạn code sau :

```
do
        X = X + 1;
        If (X==20)
                X = 0;
While (TRUE);
```

Bắt đầu với giá trị X = 0, chứng tỏ rằng giá trị X có thể vượt quá 20. Cần sửa chữa đoạn chương trình trên như thế nào để bảo đảm X không vượt quá 20 ?

Đáp án:

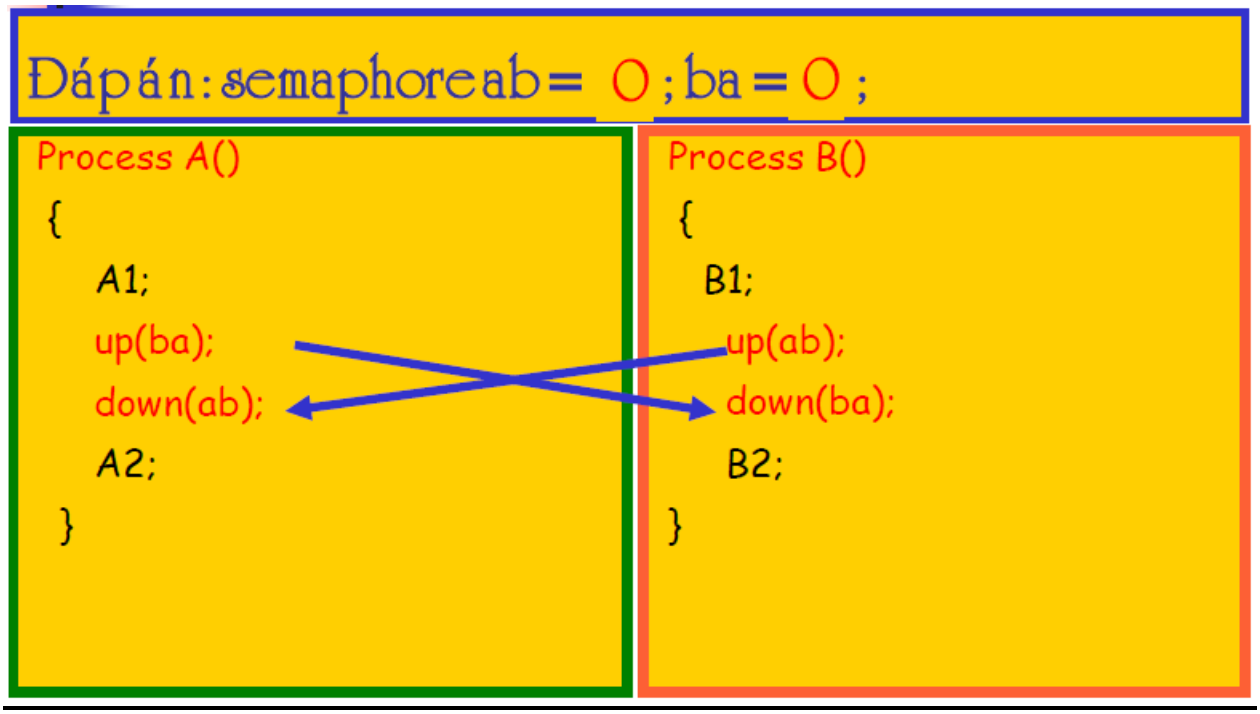
Đáp án:

```
Semaphore mutex = 1;
do
   {
      down(mutex);
      X = X +1;
      if ( X == 20) X = 0;
      up(mutex);
   }while ( TRUE );
```

2. Xét hai tiến trình xử lý đoạn chương trình sau :

process P1 { A1 ; A2 }
process P2 { B1 ; B2 }

Đồng bộ hoá hoạt động của hai tiến trình này sao cho cả A1 và B1 đều hoàn tất trước khi A2 hay B2 bắt đầu.


Đáp án: semaphore ab = 0 ; ba = 0 ;
Process A()
{
A1;
up(ba);
down(ab);
A2;
}
Process B()
{
B1;
up(ab);
down(ba);
B2;
}

Câu hỏi:

Cho mảng sau: int x[20];

Sử dụng cơ chế đồng bộ hoá là semaphore để viết code cho 3 process B,C,D cùng thực hiện đồng thời các thao tác trên mảng x thoả mãn các yêu cầu sau:

a. B chịu trách nhiệm tính tổng giá trị các phần tử mảng x có chỉ số chẵn.
b. C chịu trách nhiệm tính tổng giá trị các phần tử mảng x có chỉ số lẽ.
c. D chịu trách nhiệm tính tổng giá trị tất cả các phần tử của mảng x, dựa trên kết quả trả về của B và C.
d. Các process được khởi động cùng lúc.
e. Các process kết thúc khi xong công việc của mình, không cần chờ lẫn nhau.
f. Phải khai thác tối đa khả năng xử lý song song,chia sẻ tài nguyên dùng chung của các process.

Giải thích ngắn gọn cho giải pháp đã chọn

Lưu ý: Được phép khai báo thêm các biến cục bộ, toàn cục cần thiết để thực hiện chương trình.

Đáp án:

Giải pháp 1:

-

Đáp án : semaphore overB =0, overC =0 ;
Interger sumB, sumC = 0;

```
Process B()
 {
     for(i=0;i<9,i++) {
     sumB +=x[2*i];
     }
     up(overB);
}
```

```
Process C()
 {
     for(i=0;i<9,i++){
     sumC +=x[2*i+1];
     }
     up(overC);

}
```

```
Process D()
 {
     down(overB);
     down(overC);
     sum=sumB+sumC;

}
```

Giải pháp 2:

Đáp án : semaphore over=0;
Interger sumB, sumC = 0;

```
Process B()
{
    for(i=0;i<9,i++) {
    sumB +=x[2*i];
    }
    up(over);
}
```

```
Process C()
{
    for(i=0;i<9,i++){
    sumC +=x[2*i+1];
    }
    up(over);

}
```

```
Process D()
{
    down(over);
    down(over);
    sum=sumB+sumC;

}
```

Giải pháp 3:

Đáp án : semaphore mutex=1;
Interger sum=0;

```
Process B()
 {
      for(i=0;i<9,i++) {
      down(mutex);
      sum +=x[2*i];
      up(mutex);
      }
}
```

```
Process C()
 {
      for(i=0;i<9,i++){
      down(mutex);
      sum +=x[2*i+1];
      up(mutex);
      }
}
```

```
Process D()
 {
 }
```

Đáp án : semaphore mutex=1, over=0;
Interger sum=0;

```
Process B()
 {
      for(i=0;i<9,i++) {
      down(mutex);
      sum +=x[2*i];
      up(mutex);
      }
      up(over);
}
```

```
Process C()
 {
      for(i=0;i<9,i++){
      down(mutex);
      sum +=x[2*i+1];
      up(mutex);
      }
      up(over);
}
```

```
Process D()
 {
      down(over);
      down(over);
}
```

Xét giảipháp đồng bộ hoá sau:

```
while (TRUE)
{
      int j = 1-i;
      flag[i]= TRUE;
      turn = i;
      while (turn== j && flag[j]==TRUE);
      critical-section ();
      flag[i] = FALSE;
      Noncritical-section ();
}
```

Đây cơ phải là một giải pháp bảo đảm được độc quyền truy xuất không nếu cho 2 tiến trình cùng chạy?

Đáp án:

Không. Xét tình huống khi flag[0] =1; turn =0=>P0 vàoCS,

Nếu lúc đó flag[1]= 1 → P1 cóthể gán turn = 1 và vào luôn CS!

Sử dụng semaphore để chạy đồng bộ hoá 7 tiến trình sau:

P1: w = x1 * x2

P2: v = x3 * x4

P3: y = v * x5

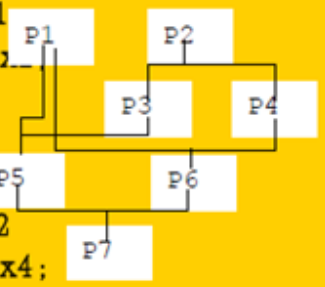
P4: z = v * x6

P5: y = w * y

P6: z = w * z

P7: ans = y + z

Đáp án :




```
process P1
{  w := x1 * x2 ;
   up(s15) ;
   up(s16) ;
}
process P2
{  v := x3 * x4 ;
   up(s23) ;
   up(s24) ;
}
process P3
{
   down(s23) ;
   y := v * x5 ;
   up(s35) ;
}
   process P4
   {
                down(s24) ;
                z := v * x ;
                up(s46) ;
   }
```

```
process P5
{
          down(s15) ;
          down(s35) ;
        y := w * y ;
        up(s57) ;
}
process P6
{
      down(s16) ;
      down(s46) ;
      z := w * z ;
      up(s67) ;
}
process P7
{
      down(s57) ;
      down(s67) ;
      ans := y + z ;
}
```

Một hãng sản xuất xe ôtô có các bộ phận hoạt động song song:

+ Bộ phận sản xuất khung xe:

```
MakeChassis() { //Sản xuất ra một khung xe
        Produce_chassis();
}
```

+ Bộ phận sản xuất bánh xe:

```
MakeTire() {            //Sản xuất ra một bánh xe
        Produce_tire();
}
```

+ Bộ phận lắp ráp: Sau khi có được 1 khung xe và 4 bánh xe thì tiến hành lắp ráp 4 bánh xe này vào khung xe:

```
Assemble(){             //Gắn 4 bánh xe vào khung xe
Put_4_tires_to_chassis();
}
```

Hãy đồng bộ hoạt động của các bộ phận trên thoả các nguyên tắc sau:

Tại mỗi thời điểm chỉ cho phép sản xuất ra 1 khung xe. Cần chờ có đủ 4 bánh xe để gắn vào khung xe hiện tại này trước khi sản xuất ra một khung xe mới.

```
Semaphore chassis=0, tire=0, wait=1;
```

```
Make_Chassis()
{
down(wait);
produce_chas();
up(chassis);
}
```

```
Make_Tire()
{
produce_Tire()
up(tire);
}
```

```
Assemble()
{
    down(chassis);
    down(tire);
    down(tire);
    down(tire);
    down(tire);
    Put_4_tires_to_chassis();
    up(wait);
}
```

Trong giai đoạn thử nghiệm, hầm đường bộ qua đèo Hải Vân chỉ cho phép các phương tiện lưu thông qua hầm với số lượng hạn chế và với những điều kiện nghiêm ngặt. Khi một phương tiện đến đầu hầm sẽ gọi hàm EnterTunnel(direction) để kiểm tra điều kiện vào hầm. Khi đã qua hầm sẽ gọi hàm ExitTunnel(direction) để báo hiệu kết thúc và rời hầm.

Giả sử hoạt động của mỗi một phương tiện được mô tả bằng tiến trình Car() sau đây:

```
Car(direction)                  //Direction xác định hướng di chuyển của phương tiện
{
        RuntoTunnel();          //Phương tiện di chuyển về phía hầm
        EnterTunnel(direction); //Đi vào hầm theo hướng direction
        PassTunnel();           //Qua hầm
        ExitTunnel(direction);  //Rời khỏi hầm theo hướng direction.
}
```

Hãy viết lại các hàm EnterTunnel(direction) và ExitTunnel(direction) kiểm soát giao thông qua hầm sao cho:

a. Tại mỗi thời điểm chỉ cho phép tối đa 3 phương tiện lưu thông qua hầm theo hướng bất kỳ.

b. Tại mỗi thời điểm chỉ cho phép tối đa 3 phương tiện lưu thông cùng hướng qua hầm.

```
Semaphore max=3;
```

```
EnterTunnel(direction)
{
    down(max);
    ...
}
```

```
ExitTunnel()
{
    ...
    up(max);
}
```