P.S. Assignment - 3

Q1) A counting semaphores s is initialized to 10. 8 wait operations followed by 6 signal operations are carried out. what is the final value of the semaphores s ?

Ans) Initially, we have semaphore value = 10

after 8 wait operations, we get the semaphore value as = 10 - 8 = 2

now, we perform 6v operation, here semaphore value will be =

= 2 + 6 = 8

Q2) Two concurrent processes P and Q are acessing their critical sections by using Boolean variables S and T as follows:

Process P	Process Q

while (true)
{
// Entry section ?
Print (1);
Print (1);
// Exit section ?
}
	while (true)
	{
	// Entry section ?
	Print(0);
	Print(0);
	// Exit section ?
	}

compare the entry section and exit section of Process P and Q with suitable semaphore operations using the two Boolean semaphores S and T. Also suggest the initial values of S and T such that execution of the processes will print the sequences 00110011....

Ans) initial value S = 0, T = 1

Process P	Process Q
while (true)	while (true)
{	{
wait (S);	wait (T);
Print (1);	Print (0);
Print (1);	Print (0);
signal (T);	signal (S);
}	}

Q3) Two concurrent processes P and Q are executing the following instructions. synchronize the execution of P and Q with suitable semaphore operations and proper initialization of the semaphore values, so that the final outcome will be in the following order:-

a) 1 3 2 4 b) 3 1 2 4

Process P	Process Q
print (1);	print (3);
print (2);	print (4);

(Ans) a) S = 1 , T = 0

Process P	Process Q
wait (S);	wait (T);
print (1);	print (3);
signal (T);	signal (S);
wait (S);	wait (T);
print (2);	print (4);
signal (T);	signal (S);

b) S = 0 , T = 1

Process P	Process Q
wait (S);	wait (T);
print (1);	print (3);
print (2);	signal (S);
signal (T);	wait (T);
	print (4);
	signal (S);

Q4) Assume the following 3 concurrent processes that use 3 binary semaphores S0, S1, S2 initialized as S0 = 1 , S1 = 0, S2 = 0. How many maximum and minimum number of times will process P0 print '0'? Justify your answer

Process P0	Process P1	Process P2
while (true) {	wait(S1);	wait (S2);
wait (S0);	signal (S0);	signal (S0);
print (0);		
signal (S1);		
signal (S2);		
}		

(Ans) minimum = 2

maximum = 3

Q5) Let 4 concurrent processes P1, P2, P3, P4 are accessing their critical sections by using Boolean semaphores S1, S2, S3 and S4. Write the entry section and exit section of all processes using the semaphores with suitable initialization, such that P1 will complete its critical section before P2 and P3, P2 and P3 will complete their critical section in any order before P4.

(Ans) $S_1 = 0$. $S2 = 0$ $S_3 = 0$ $S_4 = 0$

```
P1              P2              P3              P4
wait(S1);       wait(S2);       wait(S3);       wait(S4);
[CS]            [CS]            [CS]            [CS]
signal(S2);     signal(S3);     signal(S4);     signal(S1);
```

OR

```
P1              P2              P3              P4
wait(S1);       wait(S2);       wait(S3);       wait(S4);
[CS]            [CS]            [CS]            [CS]
signal(S3);     signal(S4);     signal(S2);     signal(S1);
```

Q6) Assume that val is an atomic integer in a Linux system. What is the value of val after the following operations have been completed?

```
atomic_set(&val, 10);
atomic_sub(8, &val);
atomic_inc(&val);
atomic_inc(&val);
atomic_add(6, &val);
atomic_sub(3, &val);
```

(Ans) As given in the question we will find the value of val,

atomic_set(&val, 10);

$\rightarrow 10$

atomic_sub(8, &val);

$\rightarrow 10 - 8 = 2$

atomic_inc(&val);

$\rightarrow 2 + 1 = 3$

atomic_inc(&val);

$\rightarrow 3 + 1 = 4$

```
atomic_add (6, &val);
    -> 6+4 = 10
atomic_sub (3, &val);
    -> 10-3 = 7
```

Q7) define the compare-and-swap hardware instruction. specify a solution to critical section problem using compare_and_swap instruction and explain how the solution will satisfy all the three requirements.

(Ans)
```
boolean CompareAndSwap (boolean *target, boolean expected,
                        boolean new)
{
    boolean rv = *target;
    if (*target == expected)
        *target = new;
    return rv;
}
```

A solution to critical section problem using compare_and_swap instruction which satisfies all three requirements.

```
do
{
  waiting [i] = T
     key = T
   while (key == T && waiting [i] == T)
       key = (&lock, F, T);
   waiting [i] = F;

   j = (i+1) % n;
   while (j != i && waiting [j] == F).

        j = (j+1) % n;
   if (i == j)
        lock = False;
        else
            waiting [i] = False.
                        RS
    } while (TRUE);
```

Q8) write a monitor solution for the bounded buffer producer consumer problem.

(Ans) Monitor PC

```
    {
      int c;
      condition full, empty;
      void put_item (item p)
      {
         if (c == N)
              full.wait();
         insert (item P);
         c = c+1;
         if (c==1)
              empty.signal(1);
      }
      void get_item()
      {
         if (c==0)
            empty.wait();
         Remove (item P);          // item P = Buffer [c];
         c = c-1;                           out = (out+1)/N
         if (c==N-1)
              full.signal();
      }
      initialization code()
      {  int c=0;
      }
    }
```

```
Producer ()
{
    while (1)
    {
        producer (item p);
        PC. put-item (item p);
    }
}
consumer()
{
    while (1)
    {
        PC. get_item ();
        consumer(item p);
    }
}
```

Q9) write a solution using semaphores for a Reader's Writers problem in which writer has higher priority than reader. Once a writer is ready, that writer performs its write as soon as possible. In other words, if a writer is waiting to access the object, no new readers may start reading.

(Ans)

```
semaphore mutex = 1          int WR = 0
int AW = 0                   int WW = 0
int AR = 0                   int OKread = 0
                             int OK write = 0
```

Reader process	Writer Process

Reader process()
 {
 while(1)
 {
	Writer process()
	 {
	 while (1)
	 {

```
wait (Mutex);
if (AW + WW == 0)
{
    signal (OK read);
    AR = AR + 1;
}
else
    WR = WR + 1
signal (Mutex);
wait (OK read);
// Reading the database
// Exit code
wait (Mutex);
AR = AR - 1;
if (AR == 0 && WW > 0)
{
    signal (OKwrite);
    AW = AW + 1;
    WW = WW - 1;
}
signal (Mutex);
}
```

```
wait (Mutex);
if (AR + AW == 0)
{
    signal (OK write);
    AW = AW + 1;
}
else
    WW = WW + 1;
signal (Mutex);
wait (OK write);
// write the database
wait (Mutex);
AW = AW - 1;
if (WW > 0)
{
    signal (OKwrite);
    AW = AW + 1;
    WW = WW - 1;
}
else
{
    while (WR > 0)
    {
        signal (OKread);
        AR = AR + 1;
        WR = WR - 1;
    }
    signal (Mutex);
}
}
```

Q10) The sleeping - Barber Problem. A barbershop consists of a waiting room with n chairs and a barber room with one barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber.

a) write a solution using semaphores to coordinate the barber & the customers.

b) write a solution using monitor to coordinate the barber & the customer.

(Ans) Barber = 0 FC = N

cus = 0 M = 1

<u>Barber code</u>

```
while (1)
{
    wait (cus);
    wait (M);
    FC++;
    signal (Barber);
    signal (M);
}
```

<u>customer code</u>

```
while (1)
{   wait (M);
    if (FC > 0) {
        FC--;
    signal (cus);
    signal (M);
    wait (Barber);
    }
    else {
        signal (M);
    }
}
```

Monitor solution :-

```
barbershop
  waiting = 0;
  customer = 0;
   barber = 0

 procedure seek - customer ()
  begin if waiting = 0
                wait (customers );
   waiting = waiting - 1 ;
   signal ( barber) ;
    end seek - costomer ;

  procedure get_haircut ()
  begin if waiting < chairs
   {
     waiting = waiting + 1;
      signal (customers ) ;
       wait (barber);
    }
   end get - haircut ;
end barbershop ;
```