CS3600 – Worksheet 04 Synchronization

Problem 1 & 2 is to understand monitor better, Problem 3 to understand the classic dining philosopher problem synchronized using semaphore and required submission for HW04.

1. While a process is executing inside the function A of the monitor m, the following calls are issued by different processes:
   m.A(); m.B(); m.B(); m.B(); m.B(); m.A(); m.A()

```
 monitor m {
          int x = 10, y = 2
          condition c
          A(){
(1)         x++
(2)         c.signal
(3)         y = x - 2 }
          B(){
(4)         if (x > 10)
(5)             x--
(6)         else{c.wait
(7)             x--} }
```

   Using the line numbers in the code, trace the sequence of instruction execution. Show any changes of x and y at the end of each instruction.

2. Complete the code below for implementing producer- consumer problem using a monitor.

```
monitor ProducerConsumer{
    condition full, empty;
    int count;

     enter()
     {
       if (count == N) wait(_______);
       put_item(item);
       count = count + 1;
       if (count == 1) signal(_______);
     }

     remove()
     {
       if (count == 0)  wait(_________);
       remove_item(item);
       count = count - 1;
       if (count == N-1) signal(________);
     }

     count = 0;
  }
  Producer();
  {
    while (TRUE)
   {
      make_item(item);
      ProducerConsumer.enter();
   }
  }
  Consumer();
  {
    while (TRUE)
    {
      ProducerConsumer.remove();
      consume_item;
   }
  }
```

This question is for Homework 04 submission

3. Trace the Dining Philosopher problem and write the flow of states using table below. If philosophers 2 , 3 and 5 try to eat. Write the states of all philosophers whenever there is a change in one philosopher state.

```
  Semaphore me= 1;                    /* for mutual exclusion */
  Semaphore s[5], all initially 0;    /* for synchronization */
  pflag[5]: {THINK, HUNGRY, EAT}, initially THINK;  /* philosopher flag */

procedure philosopher(i) //starts here
   {
while TRUE do
 {
 THINKING;
 take_chopsticks(i);
 EATING;
 drop_chopsticks(i);
 }
}
procedure take_chopsticks(i)
{
  P(me);                    /* critical section */
  pflag[i] = HUNGRY;
  test[i];
  V(me);                   /* end critical section */
  P(s[i])                  /* Eat if enabled */
}

void test(i)              /* Let phil[i] eat, if waiting */
{
if ( pflag[i] == HUNGRY && pflag[i-1] != EAT && pflag[i+1] != EAT)then
  {
    pflag[i] = EAT;
    V(s[i])
   }
}
void drop_chopsticks(int i)
{
  P(me);                  /* critical section */
  pflag[i]=THINKING;
  test(i-1);                /* Let phil. on left eat if possible */
  test(i+1);                /* Let phil. on right eat if possible */
  V(me);                   /* up critical section */
}
```

Write the change in state of philosophers [T,E,H] . Initially all are in the thinking state[T].

P[1]	P[2]	P[3]	P[4]	P[5]
T	T	T	T	T