1. Consider the following program:

```
co    < await   (x   >=   3)    x =   x - 3; >
//    < await   (x   >=   2)    x =   x - 2; >
//    < await   (x   ==   1)    x =   x + 5; >
oc
```

(a) For what initial values of x does the program terminate assuming that scheduling is weakly fair? What are the corresponding final values? (b) Write a proof outline to demonstrate your claim.

2. Suppose we have to write a busy-wait and weakly fair implementation for semaphores for a machine that has atomic increment and decrement instructions. INC(x) atomically adds 1 to integer variable x, and DEC(x) atomically subtracts 1 from integer variable x. Both INC(x) and DEC(x) return the modified value of x. Answer with an concise argument to the following questions: (a) Is the solution safe ? (b) Is the solution free from deadlock (c) Is the solution free from livelock? (d) What is wrong with it performance wise? Correct the solution in cases needed. (e) Is your solution fair after this ?

```
/* P(s): */
while (DEC(s) < 0) {
       INC(s);  #undo decrement
       }

/* V(s): */
INC (s)
```

3. Dining philosopher on a row: Five philosophers sit along a long table spending their time thinking and eating. There are six forks on the table, one on each side of every philosopher. For eating a philosopher needs exclusive access to both of the forks.

```
sem fork[6]= {1, 1, 1 , 1 , 1,  1};
process Philosopher [i = 0 to 4] {
       while (alive)  {
              P(fork[i]); P(fork[i+1]);
              eat;
              V(fork[i]); V(fork[i+1]);
              think;
       }
}
```

(a) What is bad with this solution? Demonstrate your claim with a trace. (b) Improve your solution so that it will still scale easily to an arbitrary number of N philosophers.

4. Explain the differences in synchronization mechanisms between Java synchronized class and the classical Signal-and-Wait (SW) monitor. Give an outline of how the implement SW-monitor using Java.

5. A simple roundabout (traffic circle) is a circular junction of streets where traffic flows in a one-way and one-lane circular stream around a central island. Vehicles in the roundabout have priority over the entering vehicle. A vechile may enter from any street and leave for any street. Assume that a roundabout connecting symmetrically N streets may contain at most 2N "standard size" vehicles at any time, i.e. one on each cross-road from a connecting street to the circle and one on the circle between each consecutive cross-roads. Outline a synchronization protocol which allows the cars to use the shared resource of the roundabout in way that they do not crash to each other on the cross-roads, nor on each other's back in the circle and give priority to those in the roundabout. Use either semaphores or monitors for synchronization.