Ques 1

```
monitor H2O_formation(semaphore lock) {
        int waiting_H, waiting_O, assign_H, assign_O; // initialized to 0
        condition wait_H, wait_O;
        Hydrogen() {
                wait(lock);
                waiting_H++;
                while(assign_H == 0) {
                        if (waiting_H >= 2 && waiting_O >= 1) {
                                waiting_H -= 2;
                                waiting_O -= 1;
                                assign_H += 2;
                                assign_O += 1;
                                wait_H.signal();
                                wait_O.signal();
                        }
                        else {
                                wait_H.wait();
                        }
                }
                Assign_H--;
                signal(lock);
        }
        Oxygen() {
                wait(lock);
                waiting_O++;
                while (assign_O == 0) {
                        if(waiting_H >= 2 && waiting_O >= 1) {
                                waiting_H -= 2;
```

```
                    waiting_O -= 1;
                    assign_H += 2;
                    assign_O += 1;
                    wait_H.signal();
                    wait_H.signal();
                }
                else {
                    wait_O.wait();
                }
            }
            assign_O--;
            signal(lock);
        }
}
```

Ques 2

Ques 3

Ques 4

```
int compare_and_swap(int *value, int expected, int new value) {
        int temp = *value;
        if(*value.fetch_xor(expected))
            *value = new value;
        return temp;
}
while (true) {
         while (compare and swap(&lock, 0, 1) != 0)
        ; /* do nothing */
        /* critical section */
        lock = 0;
        /* remainder section */
}
```

Ques 5

In line 6 : we signal to the next process which is in the suspended queue to enter the ready queue, otherwise the next process outside the monitor is signaled.

In line 11 : we suspend the ongoing process and add it to the end of the suspended queue and the first process in the suspended queue is given signal to start.

In line 19 : here the next process is kept into the waiting state until prompted by the signal command.

Ques 6

Starvation is possible in this case as it does not take into account a bounded waiting time. Suppose we have 4 philosophers (P1, P2, P3, P4) and 4 chopsticks, then it is possible that P2 is hungry and it checks P1(which is not eating) and P3(which is eating). It would wait for P3 to complete but it is possible that in the meantime P1 starts eating. Also this condition can go on with alternating between P1 and P3 hence not giving the chance to P2. This can cause starvation.

Ques 7

```
semaphore representative = 1;
semaphore mutex = 1;
int student_count = 0;
representative() {
        while (true) {
                wait(representative);
                if (student_count == 0) {
                        /*do other work */
                }
                else {
                        ... /* interview is going on */ ...
                        signal(representative);
                }
        }
}
student() {
```

```
        while (true) {
                wait(mutex);
                student_count++;
                if (student_count > N) {
                        break;
                }
                else {
                        signal(representative);
                        ... /* interview is performed */ ...
                        signal(mutex);
                        student_count--;
                        if (student_count == 0)
                                signal(representative);
                }
        }
}
```