```
/* Build CO2 with lock and CV */
int wC = 0; // Number of waiting C to be formed
int wO = 0; // Number of waiting O to be formed
int aC = 0; // Number of assigned C to CO2
int aO = 0; // Number of assigned O to CO2

pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t waitingC = PTHREAD_COND_INITIALIZER;
pthread_cond_t waitingO = PTHREAD_COND_INITIALIZER;

CArrives() {
        pthread_mutex_lock(&lock);
        wC = wC+1;
        pthread_mutex_unlock(&lock);
        while(aC == 0) {
                if (wO >= 2 && wC >= 1) {
                        pthread_mutex_lock(&lock);
                        MakeCO2();
                        wC = wC - 1;
                        aC = aC + 1;
                        wO = wO - 2;
                        aO = aO + 2;
                        pthread_mutex_unlock(&lock);
                        pthread_cond_signal(&waitingO);
                        pthread_cond_signal(&waitingC);
                }
                else {
                        pthread_cond_wait(&waitingC,&lock);
                }
        }
        pthread_mutex_lock(&lock);
        aC = aC - 1;
        pthread_mutex_unlock(&lock);
}

OArrives() {
        pthread_mutex_lock(&lock);
        wO = wO+1;
        pthread_mutex_unlock(&lock);
        while(aO == 0) {
                if (wO >= 2 && wC >= 1) {
                        pthread_mutex_lock(&lock);
                        MakeCO2();
                        wC = wC - 1;
```

```
            aC = aC + 1;
            wO = wO - 2;
            aO = aO + 2;
            pthread_mutex_unlock(&lock);
            pthread_cond_signal(&waitingO);
            pthread_cond_signal(&waitingC);
        }
        else {
            pthread_cond_wait(&waitingO,&lock);
        }
    }
    pthread_mutex_lock(&lock);
    aO = aO - 1;
    pthread_mutex_unlock(&lock);
}
```

```
/* Build C2H6O with lock and CV */
int wC = 0; // Number of waiting C to be formed
int wO = 0; // Number of waiting O to be formed
int wH = 0; // Number of waiting H to be formed
int aC = 0; // Number of assigned C to C2H6O
int aO = 0; // Number of assigned O to C2H6O
int aH = 0; // Number of assigned H to C2H6O

pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t waitingC = PTHREAD_COND_INITIALIZER;
pthread_cond_t waitingO = PTHREAD_COND_INITIALIZER;
pthread_cond_t waitingH = PTHREAD_COND_INITIALIZER;

CArrives() {
    pthread_mutex_lock(&lock);
    wC = wC+1;
    pthread_mutex_unlock(&lock);
    while(aC == 0) {
        if (wC >= 2 && wH >= 6 && wO >= 1) {
            pthread_mutex_lock(&lock);
            MakeC2H6O();
            wC = wC - 2;
            aC = aC + 2;
            wH = wH - 6;
            aH = aH + 6;
            wO = wO - 1;
            aO = aO + 1;
            pthread_mutex_unlock(&lock);
            pthread_cond_signal(&waitingO);
            pthread_cond_signal(&waitingC);
            pthread_cond_signal(&waitingH);
        }
        else {
            pthread_cond_wait(&waitingC,&lock);
        }
    }
    pthread_mutex_lock(&lock);
    aC = aC - 1;
    pthread_mutex_unlock(&lock);
}

OArrives() {
    pthread_mutex_lock(&lock);
    wO = wO+1;
    pthread_mutex_unlock(&lock);
    while(aO == 0) {
        if (wC >= 2 && wH >= 6 && wO >= 1) {
            pthread_mutex_lock(&lock);
            MakeC2H6O();
            wC = wC - 2;
            aC = aC + 2;
            wH = wH - 6;
            aH = aH + 6;
            wO = wO - 1;
            aO = aO + 1;
            pthread_mutex_unlock(&lock);
            pthread_cond_signal(&waitingO);
            pthread_cond_signal(&waitingC);
            pthread_cond_signal(&waitingH);
        }
        else {
            pthread_cond_wait(&waitingO,&lock);
        }
    }
    pthread_mutex_lock(&lock);
    aO = aO - 1;
    pthread_mutex_unlock(&lock);
}

HArrives() {
    pthread_mutex_lock(&lock);
    wH = wH+1;
    pthread_mutex_unlock(&lock);
    while(aH == 0) {
        if (wC >= 2 && wH >= 6 && wO >= 1) {
            pthread_mutex_lock(&lock);
            MakeC2H6O();
            wC = wC - 2;
            aC = aC + 2;
            wH = wH - 6;
            aH = aH + 6;
            wO = wO - 1;
            aO = aO + 1;
            pthread_mutex_unlock(&lock);
            pthread_cond_signal(&waitingO);
            pthread_cond_signal(&waitingC);
            pthread_cond_signal(&waitingH);
        }
        else {
            pthread_cond_wait(&waitingH,&lock);
        }
    }
    pthread_mutex_lock(&lock);
    aH = aH - 1;
    pthread_mutex_unlock(&lock);
}
```