# COMP3500: Synchronization 2 - Mutual Exclusion

**🟊: >85%, 🟊🟊: 70-85%, 🟊🟊🟊: 55-70%, 🟊🟊🟊🟊: 40-55%, 🟊🟊🟊🟊🟊: < 40%**

**Exercise 1:** Suppose there are two processes i and j.What does the following program do? Why we need flag and turn in Peterson’s solution? Please focus on the while condition. (1 minute plus 1-minute group discussion).

do {

flag[i] = true;

turn = j;

while (flag[j] && turn = = j);

critical section

flag[i] = false;

remainder section

} while (true);

**🟊🟊🟊 Exercise 1.1 (Plickers):** What does Flag[j] = false mean?

1. Process i is not in the critical section.
2. Process j is not in the critical section.
3. It is Process i’s turn now.
4. It is Process j’s turn now.

**🟊🟊 Exercise 1.2 (Plickers):** The following statements explain the meaning of turn = j . Which statement is **incorrect**?

1. The turn should be switched now.
2. It is another process’s turn now.
3. It is Process i’s turn now.
4. It is Process j’s turn now.

**🟊🟊 Exercise 2 (Plickers):** How does Peterson’s solution meet the “bounded-waiting” requirement? (1 minute)

1. Using critical section.
2. Using flag[j].
3. Using flag[i].
4. using turn = j.

**🟊🟊 Exercise 3 (Plickers):** what is **NOT** a problem with interrupt disabling? (1 minute)

1. The efficiency of execution could be noticeably degraded.
2. This approach will not work in a multiprocessor architecture.
3. This approach can’t enforce mutual exclusion.
4. This approach can’t be applied to real-time systems.

**Exercise 4:** What does the following function do? (1 minute)

boolean test\_and\_set (boolean \*target)

{

boolean rv = \*target;

\*target = TRUE;

return rv;

}

**Exercise 5:** What does the following function do? (1 minute)

int compare\_and\_swap(int \*value, int expected, int new\_value) {

int temp = \*value;

if (\*value == expected)

\*value = new\_value;

return temp;

}