1. **Project Description.** In this project you are going to implement a Prolog program to assign TAs to proctor quizzes based on their teaching schedule. There are quizzes at different timings during the week and each quiz needs a specific number of proctors. There might also be multiple quizzes during the same slot. These proctors are TAs but they cannot be assigned to a quiz on a slot that they also teach on. They also cannot be assigned to proctor a quiz on their day off. Therefore, we need a system to find any possible assignments of TAs to quizzes, verify a given assignment or output false if no assignment is possible.
   Below are the main constraints that need to be satisfied any proctoring assignment:

   -Each quiz needs a given number of proctors.

   -No TA can be assigned to proctor at the same time as a teaching slot.

   -No TA can be assigned to proctor on their day-off.

   -A TA cannot be assigned to two quizzes during the same slot.

2. **Required Predicates.** Your implementation must contain the four below predicates. Read the description of all of them before you start your implementation.

   Note that:

   1. You can add any other helper predicates you need.
   2. You can use any predefined predicates **except for `assert`** and **`retract`**.

   a) `assign_proctors(AllTAs, Quizzes, TeachingSchedule, ProctoringSchedule)` such that:

   - `AllTAs` is a list of structure representing TAs in the form `ta(Name, Day_Off)` where `Day_Off` is a the first 3 letters of the TA's day-off.
     Example: `[ta(s,sat), ta(h,tue), ta(m,thu), ta(a,sat)]`
   - `Quizzes` is a list of structure representing Quizzes in the form `quiz(Course, Day, Slot, Count)` where `Course` is a constant representing the course code that course belongs to. `Day` is a constant representing the first 3 letters of the day of the week. `Slot` is the number of the slot which can be 1, 2, 3, 4 or 5. `Count` is the number of TAs needed to be assigned to that quiz. There might be multiple quizzes in the same slot.
     Example: `[quiz(csen403, sat, 1, 5), quiz(csen401, mon, 5, 3)]`
   - `TeachingSchedule` a list of 6 day structures `day(DayName, DaySchedule)` where `DayName` is a constant representing the first 3 letters of the day of the week and `DaySchedule` is a list of the 5 slots in the day and each slot is a list of TA names that teach during that slot.
     Example: (CSEN403 course schedule)

```
[day(sat, [[], [s], [s], [s], []]),
day(sun, [[m, h], [], [s, m, h, a], [a, h], []]),
day(mon, [[h], [], [h], [h], []]),
day(tue, [[], [m], [m], [], [m]]),
day(wed, [[], [], [m], [m, a], []]),
day(thu, [[s], [a, s], [a, s], [a], []])]
```

- `ProctoringSchedule` where proctoring schedule has each quiz assigned a list of TA names to proctor it.

  Example:

  ```
  [proctors(quiz(csen403, sat, 1, 2), [m, h]),
              proctors(quiz(csen401, mon, 5, 3), [m, a, h])]
  ```

`assign_proctors/4` succeeds when all quizzes have a corresponding list of proctors of sufficient number and are all free at that time and none of them is proctoring on their day-off.

Example Query:

```
?- assign_proctors([ta(s,tue), ta(h,tue), ta(m,thu), ta(a,sat)],
                   [quiz(csen403, sun, 5, 2), quiz(csen401, mon, 2, 3)],
                   [day(sat, [[], [s], [s], [s], []]),
                   day(sun, [[m, h], [], [s, m, h], [h], []]),
                   day(mon, [[h], [], [h], [h], []]),
                   day(tue, [[], [m], [m], [], [m]]),
                   day(wed, [[], [], [m], [m], []]),
                   day(thu, [[], [s], [s], [], []])],
                   ProctoringSchedule).

ProctoringSchedule = [proctors(quiz(csen403, sun, 5, 2), [s, h]),
                        proctors(quiz(csen401, mon, 2, 3), [s, h, m])] ;
ProctoringSchedule = [proctors(quiz(csen403, sun, 5, 2), [s, h]),
                        proctors(quiz(csen401, mon, 2, 3), [s, m, h])] ;
ProctoringSchedule = [proctors(quiz(csen403, sun, 5, 2), [s, h]),
                        proctors(quiz(csen401, mon, 2, 3), [h, s, m])] ;
```

b) `free_schedule(AllTAs, TeachingSchedule, FreeSchedule)`such that:

- `AllTAs` and `TeachingSchedule` are formatted the same way as described above.
- `FreeSchedule` is formatted the same as `TeachingSchedule`.

`free_schedule/3` succeeds when `Free_Schedule` is the schedule of all free TAs on each slot. A TA is free on a slot if they have no teaching on that slot and it is not on their day off.

Example Query:

```
?- free_schedule([ta(s,tue), ta(h,sat), ta(m,thu)],
                 [day(sat, [[], [s], [s], [s], []]),
                  day(sun, [[m, h], [], [s, m, h], [h], []]),
                  day(mon, [[h], [], [h], [h], []]),
                  day(tue, [[], [m], [m], [], [m]]),
                  day(wed, [[], [], [m], [m], []]),
                  day(thu, [[], [s], [s], [], []])],
                 FreeSchedule).

FreeSchedule = [day(sat, [[s, m], [m], [m], [m], [s, m]]),
                day(sun, [[s], [s, h, m], [], [s, m], [s, h, m]]),
                day(mon, [[s, m], [s, h, m], [s, m], [s, m], [s, h, m]]),
                day(tue, [[h, m], [h], [h], [h, m], [h]]),
                day(wed, [[s, h, m], [s, h, m], [s, h], [s, h], [s, h, m]]),
                day(thu, [[s, h], [h], [h], [s, h], [s, h]])]
```

c) `assign_quizzes(Quizzes, FreeSchedule, ProctoringSchedule)` such that:

- `Quizzes` is a list of structure representing Quizzes in the form `quiz(Course, Day, Slot, Count)` where `Course` is a constant representing the course code that course belongs to. `Day` is a constant representing the first 3 letters of the day of the week. `Slot` is the number of the slot which can be 1, 2, 3, 4 or 5. `Count` is the number of TAs needed to be assigned to that quiz. There might be multiple quizzes in the same slot.
  Example: `[quiz(csen403, sat, 1, 5), quiz(csen401, mon, 5, 3)]`
- `FreeSchedule` is the schedule of when TAs are free (same format as described above).
- `ProctoringSchedule` where proctoring schedule has each quiz assigned a list of TA names to proctor it (same format as described above).

`assign_quizzes/3` succeeds when all quizzes have a corresponding list of proctors of sufficient number and are all free at that time and none of them is proctoring on their day-off. **Example Query:**

```
?- assign_quizzes([quiz(csen403, sun, 5, 2), quiz(csen401, mon, 2, 3)],
                  [day(sat, [[s, m], [m], [m], [m], [s, m]]),
                  day(sun, [[s], [s, h, m], [], [s, m], [s, h, m]]),
                  day(mon, [[s, m], [s, h, m], [s, m], [s, m], [s, h, m]]),
                  day(tue, [[h, m], [h], [h], [h, m], [h]]),
                  day(wed, [[s, h, m], [s, h, m], [s, h], [s, h], [s, h, m]]),
                  day(thu, [[s, h], [h], [h], [s, h], [s, h]])],
                  ProctoringSchedule).

ProctoringSchedule = [proctors(quiz(csen403, sun, 5, 2), [s, h]),
                        proctors(quiz(csen401, mon, 2, 3), [m, s, h])] ;
ProctoringSchedule = [proctors(quiz(csen403, sun, 5, 2), [s, m]),
                        proctors(quiz(csen401, mon, 2, 3), [m, h, s])] ;
ProctoringSchedule = [proctors(quiz(csen403, sun, 5, 2), [m, s]),
                        proctors(quiz(csen401, mon, 2, 3), [s, h, m])]
```

d) **assign_quiz(Quiz, FreeSchedule, AssignedTAs)** such that:

- **Quiz** is a single quiz structure
- **FreeSchedule** is the schedule of when TAs are free (same format as described above).
- **AssignedTAs** is a list of TA names (names only, not structures).

assign_quiz/3 succeeds when **AssignedTAs** is a list of possible TAs that can proctor **Quiz** as they are free on its slot. The number of TAs has to be equal to the number needed for that quiz and it can be in any order. If there aren't enough TAs or the TAs are not free, it fails.

Example Query:

```
?- assign_quiz(quiz(csen403, sun, 5, 2),
            [day(sat, [[s, m], [m], [m], [m], [s, m]]),
             day(sun, [[s], [s, h, m], [], [s, m], [s, h, m]]),
             day(mon, [[s, m], [s, h, m], [s, m], [s, m], [s, h, m]]),
             day(tue, [[h, m], [h], [h], [h, m], [h]]),
             day(wed, [[s, h, m], [s, h, m], [s, h], [s, h], [s, h, m]]),
             day(thu, [[s, h], [h], [h], [s, h], [s, h]])],
             AssignedTAs).

Assigned_TAs = [s, h] ;
Assigned_TAs = [h, s] ;
Assigned_TAs = [s, m] ;
Assigned_TAs = [m, s] ;
Assigned_TAs = [h, m] ;
Assigned_TAs = [m, h] ;
false.
```