TP N°1 :

Exercice 1 : les faits (facts)

1. Créer deux faits :
   - Aujourd'hui est Samedi

     (assert(aujourdhui_est samedi))
   - Demain est Dimanche
   - Hier est Vendredi
2. Vérifier si les deux faits sont bien ajoutés correctement.

   (facts)
3. Supprimer le dernier fait (numéro 3)

   (retract 3)
4. Vérifier les faits restants.

   (facts)
5. Implémenter la règle suivante :
   - Si Aujourd'hui est Samedi ALORS demain est Dimanche

     (defrule r1
     (aujourdhui_est samedi)
     =>
     (assert(demain_est dimanche))
)
6. Inférer les règles.

   (run)
7. Réinitialiser le système en supprimant tous les faits de la base des faits et les règles de la base des règles.

   (clear)

Exercice 2 : les templates (deftemplate)

1. Définir une template *personne* caractérisée par les attributs suivants :
   a. Nom : de type string et peut avoir un nom composé (de type med ali) (utiliser l'attribut cardinality)
   b. Prénom : de type string
   c. Age : de type integer
   d. Sexe : de type String et qui peut avoir comme valeur que mâle ou femelle

   (deftemplate personne
   (multislot nom (type STRING) (cardinality 1 2))
   (slot prenom (type STRING))
   (slot age (type INTEGER))
   (slot sexe (type STRING)(allowed-values "Male" "Female"))
)

2. Créer le fait de type *personne* suivant :
    a. Med ali makki, 23 ans, male

    (assert (personne(nom "Mohamed" "Ali") (prenom "Lamjed") (age 21) (sexe "Male")))
3. Créer la liste de faits, appelée groupe, de type *personne* suivante :
    a. Amani kouki, 20 ans, femelle
    b. Nizar ali, 35 ans, male
    c. Ali mongi, 22 ans, male

    (deffacts groupe
    (personne(nom "Amani") (prenom "Kouli") (age 21) (sexe "Female"))
    (personne(nom "Nizar") (prenom "Ali") (age 30) (sexe "Male"))
)
4. Ajouter la liste à la base des faits.

    (reset)
5. Vérifier la liste des faits.

    (facts)

Exercice 3 : les règles (rules)

1. Créer une première règle chercher-Nizar qui cherche la personne nizar et affiche « Bonjour Nizar mongi »

```
(defrule cherN
(personne(nom "Nizar") (prenom ?a))
=>
(printout t "Bonjour Nizar" ?a  crlf)
)
```

2. Créer une règle chercher-Age qui affiche l'âge de la personne dont le nom est amani

```
(defrule cherAge
(personne(nom "Amani") (age ?a))
=>
(printout t "l'age de amani est" ?a  crlf)
)
```

3. Créer une règle Age-Supérieur qui affiche les noms des personnes ayant un âge supérieur à 21

```
(defrule AgeSup
(personne(nom ?a) (age ?b))
(test(>= ?b 21))
=>
(printout t "Le nom est " ?a  crlf)
)
```

4. Créer une règle modifier-age qui permet de modifier l'âge d'Ali en lui ajoutant 2 ans. Que constatez-vous ?

```
(defrule ModifAge
?per<-(personne(nom "Nizar") (age ?b))
```

```
=>
(modify ?per (age (+ ?b 2)))
)
```

5. En se basant sur votre constatation, proposez une autre solution.

```
(deftemplate personne
 (multislot nom (type STRING) (cardinality 1 2))
 (slot prenom (type STRING))
 (slot age (type INTEGER))
 (slot sexe (type STRING)(allowed-values "Male" "Female"))
 (slot repeter(type SYMBOL) (default True))
)


(defrule ModifAge
?per<-(personne(nom "Nizar") (age ?b)(repeter True)
=>
(modify ?per (age (+ ?b 2))(repeter False))
)
```