```
         CLIPS (6.4.2 1/14/25)
CLIPS> (deftemplate animal
   (slot type))
CLIPS> (defrule check_animal
   (animal (type ?t&~dog))
   =>
   (printout t "Animal type is " ?t crlf))
CLIPS> (assert (animal (type cat)))
<Fact-1>
CLIPS> (run)
Animal type is cat
CLIPS> (assert (animal (type dog)))
<Fact-2>
CLIPS> (run)
CLIPS>
```

```
         CLIPS (6.4.2 1/14/25)
CLIPS> (deftemplate number
   (slot value))
CLIPS> (defrule pos_int
   (number (value ?v&:(integerp ?v) &:(> ?v 0)))
   =>
   (printout t "The number is a positive integer" crlf))
CLIPS> (assert (number (value 10)))
<Fact-1>
CLIPS> (run)
The number is a positive integer
CLIPS> (assert (number (value -10)))
<Fact-2>
CLIPS> (run)
CLIPS>
```

```
         CLIPS (6.4.2 1/14/25)
CLIPS> (deftemplate animal
   (slot type))
CLIPS> (defrule check_animal
   (animal (type ?t&:(or (eq ?t duck) (eq ?t turtle))))
   =>
   (printout t "Animal type is: " ?t crlf))
CLIPS> (assert (animal (type duck)))
<Fact-1>
CLIPS> (run)
Animal type is: duck
CLIPS> (assert (animal (type turtle)))
<Fact-2>
CLIPS> (run)
Animal type is: turtle
CLIPS> (assert (animal (type dog)))
<Fact-3>
CLIPS> (run)
CLIPS>
```

```
         CLIPS (6.4.2 1/14/25)
CLIPS> (deftemplate rect
   (slot height)
   (slot width))
CLIPS> (defrule perimeter
   (rect (height ?h) (width ?w))
   =>
   (bind ?p (* 2 (+ ?h ?w)))
   (printout t "Perimeter of the rectangle is: " ?p crlf))
CLIPS> (assert (rect (height 5) (width 3)))
<Fact-1>
CLIPS> (run)
Perimeter of the rectangle is: 16
CLIPS>
```

```
         CLIPS (6.4.2 1/14/25)
CLIPS> (deftemplate animal
   (slot type))
CLIPS> (defrule check_animal
   (animal (type ?t&:(and (neq ?t dog) (neq ?t cat))))
   =>
   (printout t "Animal type is: " ?t crlf))
CLIPS> (assert (animal (type duck)))
<Fact-1>
CLIPS> (run)
Animal type is: duck
CLIPS> (assert (animal (type cat)))
<Fact-2>
CLIPS> (run)
CLIPS> (assert (animal (type dog)))
<Fact-3>
CLIPS> (run)
CLIPS>
```

```
         CLIPS (6.4.2 1/14/25)
CLIPS> (deftemplate number
   (slot value))
CLIPS> (defrule Odd_number
   (number (value ?v))
   (test (= (mod ?v 2) 1))
   =>
   (printout t "The number is odd" crlf))
CLIPS> (assert (number (value 7)))
<Fact-1>
CLIPS> (run)
The number is odd
CLIPS> (assert (number (value 4)))
<Fact-2>
CLIPS> (run)
CLIPS>
```

```
         CLIPS (6.4.2 1/14/25)
CLIPS> (deftemplate person
   (slot hair-color))
CLIPS> (defrule check_color
   (person (hair-color ?color&:(neq ?color "black") &:(neq ?color "brown")))
   =>
   (printout t "The person's hair color is: " ?color crlf))
CLIPS> (assert (person (hair-color "red")))
<Fact-1>
CLIPS> (run)
The person's hair color is: red
CLIPS> (assert (person (hair-color "black")))
<Fact-2>
CLIPS> (run)
CLIPS> (assert (person (hair-color "brown")))
<Fact-3>
CLIPS> (run)
CLIPS>
```

```
         CLIPS (6.4.2 1/14/25)
CLIPS> (deftemplate person
   (multislot name
      (type SYMBOL STRING)
      (cardinality 2 4))
   (slot age
      (type INTEGER)
      (range 20 25)))
CLIPS> (assert (person (name "Ali" Ahmed) (age 23)))
<Fact-1>
CLIPS> (assert (person (name Ahmed) (age 23)))
[CSTRNCHK1] Literal slot values found in the 'assert' command does not satisfy the cardinality restrictions for slot 'name'.
CLIPS> (assert (person (name "Ali" Ahmed) (age 9)))
[CSTRNCHK1] A literal slot value found in the 'assert' command does not fall in the allowed range 20 to 25 for slot 'age'.
CLIPS> (assert (person (name "Ali") (age 9)))
[CSTRNCHK1] Literal slot values found in the 'assert' command does not satisfy the cardinality restrictions for slot 'name'.
CLIPS> (facts)
f-1     (person (name "Ali" Ahmed) (age 23))
For a total of 1 fact.
CLIPS>
```