Laborator 9
Introducere in Clips

1. Scrieti un program care citeste un numar natural n si scrie toate numerel naturale dintre 0 si n.

```
(deffacts fapte (contor 0))

(defrule citire
        (not (n ?))
=>
        (printout t "n = ")
        (assert (n (+ 1 (read))))
)

(defrule scrie
        (n ?nr)
        (not (contor ?nr))
        ?f <- (contor ?cnt)
=>
        (retract ?f)
        (assert (contor (+ ?cnt 1)))
        (printout t ?cnt crlf)
)
```

2. Scrieti un program care citeste un numar natural n si afiseaza factorialul fiecarui numar natural <= n. Dupa fiecare factorial afisat utilizatorul este intrebat daca executia continua.

```
(deffacts fapte (raspuns da) (factorial 1) (cont) (contor 1))
```

```
(defrule citire
        (not (n ?))
=>
        (printout t "n = " )
        (assert (n (+ 1 (read))))
)

(defrule fact
        (n ?nr)
        ?f1 <- (raspuns da)
        ?f2 <- (contor ?cnt)
        ?f3 <- (factorial ?rez)
=>
        (retract ?f1)
        (retract ?f2)
        (retract ?f3)
        (assert (contor (+ ?cnt 1)))
        (assert (factorial (* ?cnt ?rez)))
        (printout t ?cnt "! = " (* ?cnt ?rez) crlf)
)

(defrule continuare
        (n ?nr)
        (not (contor ?nr))
        (not (raspuns ?))
=>
        (assert (cont1))
)
```

```
(defrule raspuns
        ?f <- (cont1)
=>
        (retract ?f)
        (printout t "Continuati (da/nu)? ")
        (assert (raspuns (read)))
)
```

3. Scrieti un program care sa rezolve problema de la sfarsitul primului tutorial (se vor folosi reguli de tipul ultimei reguli din tutorialul 2)

```
(deffacts fapte (init))

(defrule starterTurning?
        ?f1 <- (init)
=>
        (retract ?f1)
        (printout t "Starter turning? (y/n): ")
        (assert (starterTurning (read)))
)

(defrule gotAnyPetrol?
        ?f2 <- (starterTurning y)
=>
        (retract ?f2)
        (printout t "Got any petrol? (y/n): ")
        (assert (gotAnyPetrol (read)))
)
```

```
(defrule callTheAA
        ?f3 <- (gotAnyPetrol y)
=>
        (printout t "Call the AA." crlf)
)

(defrule buySome
        ?f4 <- (gotAnyPetrol n)
=>
        (printout t "Buy some!" crlf)
)

(defrule lightsWorking?
        ?f5 <- (starterTurning n)
=>
        (retract ?f5)
        (printout t "Lights working? (y/n): ")
        (assert (lightsWorking (read)))
)

(defrule solenoidClick?
        ?f6 <- (lightsWorking y)
=>
        (retract ?f6)
        (printout t "Solenoid click? (y/n): ")
        (assert (solenoidClick (read)))
)

(defrule terminalsClean?
        ?f7 <- (solenoidClick y)
=>
```

```
        (retract ?f7)
        (printout t "Terminals clean? (y/n): ")
        (assert (terminalsClean (read)))
)

(defrule replaceStarter
        ?f8 <- (terminalsClean y)
=>
        (retract ?f8)
        (printout t "Replace Starter" crlf)
)

(defrule cleanTerminals
        ?f9 <- (terminalsClean n)
=>
        (retract ?f9)
        (printout t "Clean Terminals" crlf)
)

(defrule solenoidFuseOK?
        ?f10 <- (solenoidClick n)
=>
        (retract ?f10)
        (printout t "Solenoid fuse OK? (y/n): ")
        (assert (solenoidFuseOK (read)))
)

(defrule replaceSolenoid
        ?f11 <- (solenoidFuseOK y)
=>
        (retract ?f11)
```

```
        (printout t "Replace Solenoid" crlf)
)

(defrule replaceFuse
        ?f12 <- (solenoidFuseOK n)
=>
        (retract ?f12)
        (printout t "Replace Fuse" crlf)
)

(defrule chargeBattery
        ?f13 <- (lightsWorking n)
=>
        (retract ?f13)
        (printout t "Charge Battery" crlf)
)
```