1. Instrukcja (loop-for-count ...)

Funkcja `(loop-for-count)` w języku CLIPS pozwala na proste iteracje. Jej składnia jest następująca:

```
(loop-for-count     <zakres-iteracji>   (<akcje>))
```

przy czym <zakres-iteracji> może przyjmować dwie postacie:

```
<zakres-iteracji> := <indeks-koncowy>
```

lub:

```
<zakres-iteracji> := (?zmienna <indeks-poczatkowy> <indeks-koncowy>)
```

przy czym `<indeks-poczatkowy>` i `<indeks-koncowy>` muszą być liczbami całkowitymi.

W przypadku pierwszym przykładowe użycie funkcji (`loop-for-count`) może być następujące:

a)
```
(loop-for-count 2
     (printout t "Podwójna iteracja" crlf))
```

Efekt działania:
```
Podwójna iteracja
Podwójna iteracja
```

b)
```
(bind ?x 2)
(loop-for-count ?x
     (printout t „OK" crlf))
```

Efekt działania:
```
OK
OK
```

c)
```
(defglobal ?*x* = 2)
(loop-for-count ?*x*
     (printout t "OK" crlf))
```

Efekt działania:
```
OK
OK
```

d)

```
(loop-for-count (?a ?*x* 4)            ; ?*x* = 2
     (printout t ?a crlf))
```

Efekt działania:

```
2
3
4
```

Funkcje `(loop-for-count ...)` mogą być zagnieżdżone. Druga z możliwości tj. rozszerzona deklaracja `<zakresu-iteracji>`, wraz z zagnieżdżeniem iteracji może być zilustrowana następująco:

```
(loop-for-count  (?a ?*x* 4)  ;?*x* = 2
     (loop-for-count (?b 1 3)
     (printout t ?a " " ?b crlf)))
```

Efekt działania:

```
2    1
2    2
2    3
3    1
3    2
3    3
4    1
4    2
4    3
```

Przykład (pętla pojedyncza – obliczanie silni):

```
(defrule start
     (initial-fact)
=>
     (printout t "Program oblicza wartosc silni z n" crlf)
     (printout t "Podaj n = ")
     (bind ?n (read))
     (assert (n ?n)))

(defrule licz
     (n ?n)
=>
     (bind ?silnia 1)
     (loop-for-count (?i 1 ?n)
     (bind ?silnia (* ?silnia ?i)))
     (assert (silnia ?silnia)))

(defrule wypisz
     (silnia ?silnia)
=>
     (printout t "Silnia = " ?silnia crlf))
```

2. Zadania

a) napisać program (przy użyciu instrukcji loop-for-count) wypisujący kolejno liczby od 1 do zadanej wartości *n*
np. po podaniu n=7, program powinien wypisać:

```
1
2
3
4
5
6
7
```

b) napisać program wypisujący na ekranie tabliczkę mnożenia postaci:

```
1    2    3    4    5    6    7    8    9    10
2    4    6    8   10   12   14   16   18    20
3    6    9   12   15   18   21   24   27    30
.....
.....
9   18   27   36   45   54   63   72   81    90
10  20   30   40   50   60   70   80   90   100
```

c) napisać program wypisujący na ekranie macierz trójkątną górną o wymiarach n na n, złożoną z jedynek
np. dla n=6, program powinien wyświetlić

```
1    1    1    1    1    1
1    1    1    1    1    0
1    1    1    1    0    0
1    1    1    0    0    0
1    1    0    0    0    0
1    0    0    0    0    0
```

d) zmodyfikować program z punktu 2c, aby program wyświetlał macierz jednostkową o zadanym przez użytkownika rozmiarze n

np. dla n=6, program powinien wyświetlić

```
0    0    0    0    0    1
0    0    0    0    1    0
0    0    0    1    0    0
0    0    1    0    0    0
0    1    0    0    0    0
1    0    0    0    0    0
```

e) napisać program obliczający (przy użyciu instrukcji loop-for-count) n-ty wyraz ciągu Fibonacciego

$F_0 = 1;\ F_1 = 2;\ F_2 = 3;\ F_3 = 5;\ \ldots$
$F_n = F_{n-1} + F_{n-2}$

np. po podaniu n=6, program powinien podać liczbę 21