

# Metody programowania 2014

## Lista zadań nr 4

Na zajęcia 25–27 marca 2014

**Zadanie 1 (1 pkt).** Zaprogramuj w Prologu predykat `length/2`, który działa poprawnie we wszystkich trybach użycia, w szczególności nie zapętla się dla celu `length(X, 5)` (ten cel powinien być spełniony dokładnie na jeden sposób, a pod `X` powinna zostać podstawiona lista `[_,-,-,-,-]`).

**Zadanie 2 (1 pkt).** Niech predykat `connection/2` (zdefiniowany przez zbiór faktów) oznacza, że istnieje bezpośrednie połączenie między dwoma miastami. Zdefiniuj predykat `trip/3`, który znajduje połączenie między dwoma miastami z dowolną liczbą przesiadek i nie zapętla się nawet wówczas, gdy w grafie połączeń są cykle. Pierwsze dwa parametry tego predykatu (wejściowe), to miasta początkowe i końcowe. Trzeci parametr (wyjściowy), to lista miast od początkowego do końcowego, przez które przebiega podróż, np.:

```
?- trip(gliwice, warszawa, T).
T = [gliwice, wroclaw, warszawa] ;
T = [gliwice, wroclaw, katowice, warszawa] ;
false.
```

**Wskazówka:** zdefiniuj w pierw predykat `trip/4`, którego dodatkowy parametr jest „akumulatorem” — listą zawierającą już odwiedzone miasta. Użyj predykatu `member/2` by sprawdzić, czy miasto, do którego chcemy przejść, było już wcześniej odwiedzone. Ścieżkę buduj od końca ku początkowi, by uniknąć konieczności odwracania listy po znalezieniu rozwiązania.

**Zadanie 3 (1 pkt).** Rozważmy binarne rozwinięcia liczb naturalnych, takie, że binarnym rozwinięciem liczby 0 jest ciąg złożony z pojedynczej cyfry 0, a rozwinięcia liczb dodatnich nie zawierają zer nieznaczących (zatem ich najbardziej znaczącą cyfrą zawsze jest 1). Napisz w Prologu predykaty `bin/1` oraz `rbin/1` generujące binarne rozwinięcia kolejnych liczb naturalnych reprezentowane w postaci list literałów 0 i 1. Predykat `bin` powinien tworzyć listy cyfr dwójkowych w kolejności od najbardziej do najmniej znaczącej pozycji, `rbin` zaś — odwrotnie. Przeciętna liczba wnioskowań pomiędzy kolejnymi sukcesami powinna być ograniczona przez stałą (o to, ile wynosi ta stała, możecie zapytać artystkę uliczną Magdę z zajęć wyrównawczych z matematyki<sup>1</sup>), nie wolno więc kopiować ani odwracać generowanych list. Nie wolno używać żadnych predykatów standardowych. Wolno definiować własne predykaty pomocnicze. Pomimo iż sformułowanie zadania odwołuje się do pojęcia liczby, nie należy używać prologowej arytmetyki.

Wynikiem zapytania `bin(X)` powinien być ciąg odpowiedzi:

<sup>1</sup>Dopiero po zakończeniu wrześniowego kursu okazało się, że Magda, to tylko pseudonim artystyczny, a w rzeczywistości artystka pochodzi z zachodniej granicy i ma na imię Tania (wskazówka: średnio 2 zł. za występ).

```
X = [0];
X = [1];
X = [1,0];
X = [1,1];
X = [1,0,0];
X = [1,0,1];
...
```

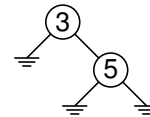
zaś wynikiem zapytania `rbin(X)` powinien być ciąg odpowiedzi:

```
X = [0];
X = [1];
X = [0,1];
X = [1,1];
X = [0,0,1];
X = [1,0,1];
...
```

**Zadanie 4 (1 pkt).** Drzewa binarne o etykietowanych wierzchołkach wewnętrznych zapisujemy w Prologu używając funktora `node/3` do reprezentowania wierzchołków wewnętrznych i atomu `leaf/0` do reprezentowania liści. Dla przykładu struktura:

```
node(leaf, 3, node(leaf, 5, leaf))
```

reprezentuje drzewo:



Zaprogramuj predykaty `mirror/2` — tworzący lustrzane odbicie drzewa oraz `flatten/2` — tworzący listę etykiet drzewa w porządku infiksowym.

**Zadanie 5 (1 pkt).** Binarne drzewa poszukiwań będziemy zapisywać podobnie, jak w poprzednim zadaniu. Zaprogramuj predykat `insert/3` — wstawiający element do drzewa (z powtórzeniami). Zaprogramuj następnie algorytm *Tre-sort*, który wstawia wszystkie elementy listy do drzewa poszukiwań i następnie spłaszcza drzewo otrzymując posortowaną listę.

**Zadanie 6 (1 pkt).** Napisz zapytanie prologowe, które rozwiąże następującą łamigłówkę:

		U	S	A
+		U	S	S
<hr/>				
=	P	E	A	C

Zapytanie powinno zawierać zmienne `A`, `C`, `E`, `P`, `R`, `S` i `U` oraz powinno sprawdzić wszystkie możliwe podstawienia pod te zmienne parami różnych cyfr 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9. Cyfry podstawione pod zmienne `U` i `P` nie mogą być zerem. W zapytaniu możesz użyć następujących predykatów standardowych:

`permutation/2`, `length/2`, `is/2`, `\=/2`,

predykatu `concat_number/2` z listy 3 i predykatu `sublist/2` z listy 2. Zapytanie powinno zwrócić dokładnie jedną odpowiedź, bez powtórzeń:

?- zapytanie (pominięte).

A = 2,

C = 7,

E = 0,

P = 1,

R = 8,

S = 3,

U = 9;

false.

?-

**Zadanie 7 (1 pkt).** Zaprogramuj predykat `revall/2` odwracający listę wraz z podlistami (tutaj przez podlistę rozumiemy element listy będący listą):

?- `revall([1,[2,3,[4,5],6,[7,[8,9]]]], X).`

`X = [[[[9,8],7],6,[5,4],3,2],1]`

Zadbaj o to, by Twój predykat nie generował nieużytków!