Kurs języka Prolog 2019

Lista zadań nr 4

Na zajęcia 27 marca 2019

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

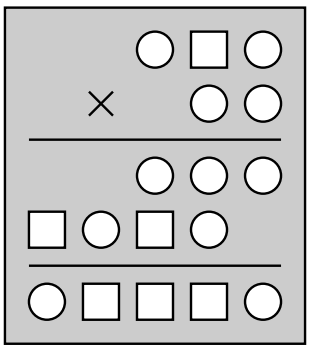
```
?- revall([1,[2,3,[4,5],6,[7,[8,9]]]], X).
X = [[[[9,8],7],6,[5,4],3,2],1]
```

Zadbaj o to, by Twój predykat nie generował nieużytków!

Zadanie 2 (1 pkt). Napisz predykat `flatten/2`, który „spłaszcza" listę:

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

Zadanie 3 (2 pkt). Napisz w Prologu program rozwiązujący zadania takie jak to, które pochodzi z numeru 11/2000 miesięcznika „Wiedza i Życie": *Wpisz w kwadraty poniższej planszy cyfry parzyste (0, 2, 4, 6, 8), w koła zaś nieparzyste (1, 3, 5, 7, 9) tak, by otrzymać poprawny zapis mnożenia.*



Dane do programu (opis problemu) należy podać w postaci listy list atomów `s` i `c` reprezentujących kwadraty i koła. Np. dane do problemu z obrazka są następujące:

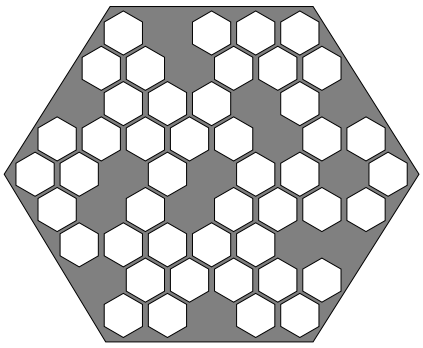
```
[[c,s,c], [c,c], [c,c,c], [s,c,s,c], [c,s,s,s,c]]
```

Zadanie 4 (2 pkt). Zaprogramuj predykat rozwiązujący zadanie ustawienia N hetmanów na szachownicy $N \times N$ tak, by wzajemnie się nie szachowały. Rozwiązanie zwracaj w postaci N-elementowej listy liczb: lista $[y_1, \ldots, y_n]$ opisuje ustawienie, w którym hetmany stoją na polach o współrzędnych (x, y_x) dla $x = 1, \ldots, N$.

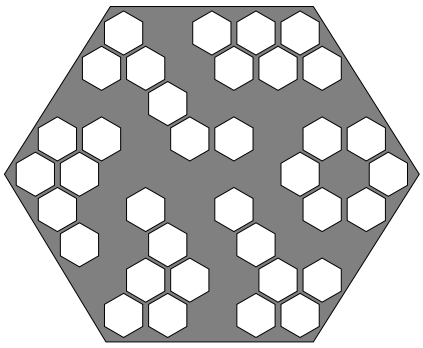
Zadanie 5 (2 pkt). Zaprogramuj predykat rozwiązujący problem skoczka na szachownicy rozmiaru $N \times M$: skoczek startuje z podanego pola (x_1, y_1) i ma wykonać $N \cdot M - 1$ ruchów tak, by odwiedzić każde pole (na żadnym polu nie może znatem znaleźć się dwukrotnie). Zgodnie z zasadami ruchu skoczka pole $(x', y') = (x + \Delta x, y + \Delta y)$ jest następnikiem pola (x, y), jeśli leży na planszy (tj. $1 \leq x' \leq N$ i $1 \leq y' \leq M$) oraz $|\Delta x| + |\Delta y| = 3$ i $-2 \leq \Delta x, \Delta y \leq 2$. Wybierz najprostszą reprezentację stanu obliczeń, w której przebyta droga jest reprezentowana w postaci listy współrzędnych, a dopuszczalne kontynuacje wyznacza się sprawdzając za pomocą predykatu `member/2`, czy pole docelowe nie występuje na dotychczasowej drodze.

Zadanie 6 (4 pkt). Problem skoczka na szachownicy $N \times N$ jest dla tego samego N *znacznie* trudniejszy obliczeniowo, niż problem N hetmanów. W pierwszym przypadku przestrzeń przeszukiwania jest rzędu $\binom{N^2}{N}$, ale z łatwością zmniejszamy ją do $N!$. W drugim przypadku przestrzeń jest rzędu c^{N^2}, gdzie c jest pewną stałą pomiędzy 2 i 8. Obliczenie rozważane w zadaniu 5 jest czasochłonne: podczas przeszukiwania za każdym razem generujemy dla danego pola potencjalne kontynuacje i dla każdej z nich sprawdzamy, czy nie była już odwiedzona. Lepiej byłoby reprezentować planszę w postaci $N \times M$-wierzchołkowego grafu, w którym krawędzie łączą te pary pól, dla których skoczek może wykonać pojedynczy ruch przechodząc z pierwszego z nich na drugie. Dzięki temu potencjalne kontynuacje drogi zostaną wyliczone dla każdego pola jednokrotnie w fazie preprocessingu i umieszczone np. na liście następników. Taka optymalizacja przyspieszy nieznacznie program, ale pozwoli też na implementację znacznie bardziej istotnego ulepszenia: zamiast dla każdej kontynuacji sprawdzać w fazie przeszukiwania, czy nie występuje już na ścieżce, możemy z każdym polem planszy związać dodatkową zmienną. Jeśli ta zmienna jest nieukonkretniona, to dane pole nie było jeszcze odwiedzone. Jeśli jest natomiast ukonkretniona, to wskazuje na poprzednika danego pola na budowanej ścieżce. Obliczenie predykatu member na liście kilkudziesięciu elementów zastępujemy zatem pojedynczym wywołaniem predykatu `var/1`. Zaimplementuj takie rozwiązanie. Przyjmij przy tym, że każde pole planszy jest reprezentowane za pomocą struktury `field/3`, w której pierwszy argument zawiera współrzędne pola, drugi — listę jego następników, a trzeci jest omówioną wyżej nieukonkretnioną zmienną. Porównaj szybkość tego programu i programu rozważanego w zadaniu 5.

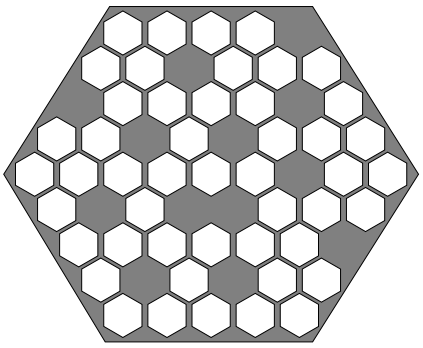
Zadanie 7 (8 pkt). Napisz w Prologu program rozwiązujący zadania takie jak to, które pochodzi z numeru 3/2000 miesięcznika „Wiedza i Życie": *Miodowe wyspy:* Usuń [w oryginale *zaczerń*] *niektóre* sześciokąty [zwane w oryginale *sześcianikami*] *tak, aby* pozostałe [w oryginale *żółte*] *utworzyły sześć „wysp". Każda wyspa powinna składać się z sześciu pól i nie może dotykać do żadnej z pozostałych wysp (jak w zamieszczonym przykładzie z rozwiązaniem).*



Przykład 1



Rozwiązanie przykładu 1



Przykład 2

Rozmiar planszy, liczba i rozmiar wysp oraz położenie usuniętych początkowo sześciokątów powinny być danymi dla Twojego programu.

Uwaga. „Puzeland" Marka Penszki, to kopalnia wielu ciekawych zadań, które często posiadają bardzo eleganckie rozwiązania w Prologu, zob. `http://archiwum.wiz.pl/tematy/temat62.asp`.