

SHIFTLEFT

shiftleft ([#|T], L2):-
conc(T, [#], L2).

shiftleft (L1, L2):-

L1 = [#|T],

* generični, lepotni

1. conc (T, H, L2).
* generični

(*)

2. nepovezava L2 iz conc in L2, ki se vrača v glavi...
(tole je idealna za AI)

3. Tulej ~~rekurzivna~~ ^{rekurzivna} pogoja pravstavnosti in. (zakaj se skrivajo conc/3)
A poskušamo odgovoriti, da v stavku ni ne rekurzije in zagona od je to majhno "povsem" nerekurzivno stavko...
kaj pa bi bil namig?
(morda kar kakšen iz spletnih namigov?)

4. ~~L2 =~~ L2 = [#|T]

Postkus narediti: karakterje, ki lahko konca na dva (obliki) načina:
(a) naprečno reverse (glej do konca in potem kajto na koncu)
(b) vrzajo esuklivesipremanje seznam (glej do konca in potem kajto na začetku)

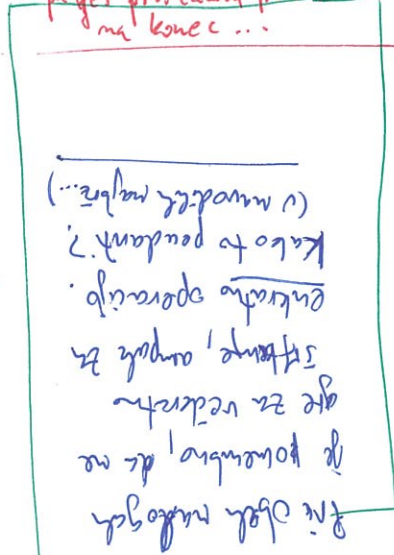
2. shiftleft (L1, L2):-
shiftleft (L2, L1).

2. shiftleft (L1, [#|L2]):-
conc (L2, [#], L1).

> kaj pa tale (pravilna) rešitev?

shiftleft (X, [X]).
shiftleft ([H1, H2 | T], [H2 | R]):-
shiftleft ([H1 | T], R).

"prijeti prvi element po seznamu na konec..."



4. L = [T | H] (isto kot pri shiftleft #4)

3. Tudi kn in rekurzivne pogoja... glej #3 pri shiftleft

2. nepovezava L2 in L2, ter L1 in L1, (u) idealno za AI.

1. "kaj mi ob, ker samo prave zaloge, da vidim, da ga ne vrže stran (ne dobit T)

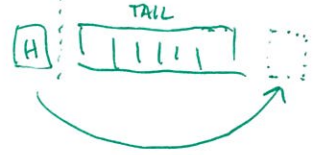
1. naprečno cut/2

zauzavost:
shiftleft je sveds ravnaj operacij od shiftleft, korig. vsiljen #2 (->)

spletni nasveti

2. Preprosto! - v seznamu seznam prvi element in ga dodamo (v preostanku) na konec. računamo T

1.



3.

"če seznam je seznam L nastavljen iz glave H in repa T in če H dodamo na konec T, potem je rezultat seznam "shiftan" o leveo".

3. "če je E zaloge! element seznam L in je T seznam brez tega elementa, torej potem je [E | T] ob "shiftanju" seznam L"

> več: kako določiti zaloge? element? (in preostanek)

2. V seznamu posleži ter ga preostanek dan in seznam stran zaloge! element na začetku.



1.

SHIFTRIGHT

SUBLIST * (to je v bistvu "tricky" uporaba conc/3)

osnovni splošni nasveti

sublist (L, SL):-
 conc(-, L1, L),
 conc(SL, -, L1).

① v bistvu ni (prav) rekurzije

HINT: "sublist lahko definirati posebej nerekurzivno" ~ use conc/3

* glavnica: delectivnej, da je uporabnik uporablja rekurzijo!

(Ta rešitev večkrat ponovi prazen seznam [] kot rešitev, a mas to ne moti preveč.)

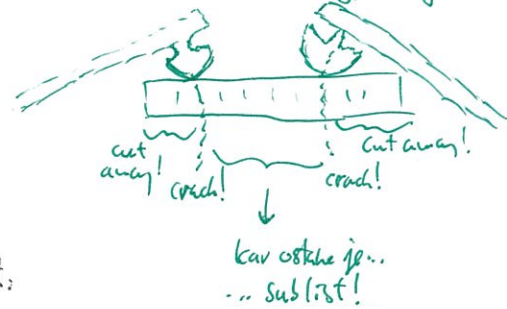
* ni problem začeti, good! (uporabljajo mit conc/3, or append/3!)

② Ko enkrat Student vognute sam ali vidi nasvet o reševanju s conc/3, se problem prevede na pravilno "večans" spremenljivke...
 ... ta je idealen case za AI. mačin od tunc!
 (najbrž preveč kombinacij (napačnost), da bi jih bilo smiselno koriti eno po eno...)

* words best zanimivost: ali je do tu že pasna ideja kalcitriranja oz. vračunja različnih vrstev? (vedo)

① Najprej razloži kaj sploh malo je zahteva... ne podrobnico, ampak podseznan (ki delavniki se držijo skupaj).

② visual hint: "sekaj, sekaj"!!



→ več: Kako "odsekati" delo strukture?
 spredaj: conc(-, ostanki, seznam)
 zadaj: conc(ostanki, -, seznam)

③ "Če ^{delamo} začeli strukturo L odsekamo nekaj na začetku in potem se nekaj na koncu, mi ostane ravno neh podstruktura..."