

Prevodioci i interpretatori - januar 2010.

1. Konstruisati MDKA za izraz $(ba*b)+ba$ i zatim napisati C program koji proverava da li uneta niska pripada ovom jeziku.
2. LL(1) gramatikom opisati deklaracije procedura i funkcija u PASCAL-u, a zatim napraviti perl skript koji tabličnom simulacijom potisnog automata proverava da li je na ulazu zadata ispravna deklaracija. Npr.

```
procedure print(var j: integer);  
function next(k: integer, l: real): integer;
```

3. Implementirati interpretator koji sračunava vrednosti izraza u kojima se javljaju prirodni brojevi i liste prirodnih brojeva. Nad brojevima je moguće vršiti sve četiri aritmetičke operacije, pri čemu je deljenje celobrojno, a oduzimanje većeg broja od manjeg daje 0. Funkcija Suc računa sledbenik broja. Npr.

Suc(5);	6
1 + 2*3;	7
3 - 1;	2
3 - 2*2;	0

Liste se zapisuju kao $[a_1, \dots, a_n]$. Operator # dopisuje element na početak liste, dok se operatorom @ nadovezuju dve liste. Npr.

[1, 2, 3];	[1, 2, 3]
1 # [];	[1]
[1, 2] @ [1+2, 4];	[1, 2, 3, 4]

Dopušteno je i korišćenje let vezivanja. Npr.

let x = 3 in 2*x;	6
let l1 = [1, 2]; l2 = [3, 4] in l1 @ l2;	[1, 2, 3, 4]

Nad listama su definisane funkcije hd koja izdvaja glavu (prvi element) liste, tl koja izdvaja rep liste, operator ! koji izdvaja n-ti element (brojanje počinje od 1), rev koja obrće listu i sort koja sortira listu rastuće. Npr.

hd([1, 2, 3]);	1
tl([1, 2, 3]);	[2, 3]
[6, 7, 8, 9] ! 2;	7
rev([1, 2, 3]);	[3, 2, 1]
sort([7, 9, 8, 6]);	[6, 7, 8, 9]

Moguće je definisanje funkcija primitivnom rekurzijom (nad brojevima i nad listama) i njihovo korišćenje u izrazima. Npr.

```
primrec sum where  
  sum(0) = 0;  
  | sum(Suc(n)) = (Suc n) + sum n;
```

sum(5);	15
---------	----

```
primrec power where  
  power(x, 0) = 1;  
  | power(x, (Suc n)) = x * power(x, n);
```

power(2, 3);	8
--------------	---

```
primrec length where  
  length([]) = 0;  
  | length(h#t) = 1 + length(t);
```

length([3, 7, 2]);	3
--------------------	---