

```

1 /* -*- C++ -*-
2 * $Date: 2011-09-20 14:02:53 +0200 (k, 20 szept 2011) $
3 * Copyright (C) 2011, BME Declarative Programming Course
4 * 2011.09.16. Gyakorlat - megoldasok
5 */
6
7 #include "ceklah"
8
9 //////////////////////////////////////////////////
10 // megoldasok
11
12 // 1.
13 int csupa01(const int N) {
14     if (N == 0) return 1;
15     if (N % 10 >= 2) return 0;
16     return csupa01(N/10);
17 }
18
19 // 2.
20 // osztok(N,D) az N szam D-nel nem kisebb osztoinak szama
21 int osztok(const int N, const int D) {
22     if (N < D) return 0;
23     const int Ez = N % D == 0;
24     return Ez + osztok(N, D+1);
25 }
26 int osztok(const int N) {
27     return osztok(N, 1);
28 }
29
30 // 3.
31 // A es B legnagyobb olyan kozos osztoja, ami nem nagyobb D-nel
32 int lnko(const int A, const int B, const int D) {
33     // if (A % D == 0)
34     //     if (B % D == 0)
35     //         return D;
36     // if (A % D + B % D == 0)
37     //     return D;
38     return lnko(A, B, D-1);
39 }
40 int lnko(const int A, const int B) {
41     return lnko(A, B, A);
42 }
43
44 // 4.
45 int lnko2(const int A, const int B) {
46     if (B == 0) return A;
47     return lnko2(B, A % B);
48 }
49
50 // 5.
51 int length(const list L) {
52     if (L == nil) return 0;
53     return 1 + length(tl(L));
54 }
55
56 // 5.*
57 // listaHossza_i(L, H) az L lista hossza + H
58 int lengthi(const list L, const int H) {
59     if (L == nil) return H;
60     return lengthi(tl(L), H + 1);
61 }
62 int lengthi(const list L) {
63     return lengthi(L, 0);
64 }
65
66 // 6.
67 list lista_noveltje(const list L) {
68     if (L == nil) return nil;
69     return cons(hd(L) + 1, lista_noveltje(tl(L)));
70 }

```

```

71
72 // 7.
73 // L nem lehet ures!
74 int last(const list L) {
75     if (tl(L) == nil) return hd(L);
76     return last(tl(L));
77 }
78
79 // 8.
80 list insert_nth(const list L, const int N, const int E) {
81     if (N == 0) return cons(E, L);
82     return cons(hd(L), insert_nth(tl(L), N - 1, E));
83 }
84
85 // 9.
86 int nthl(const list L, const int N) {
87     if (N == 1) return hd(L);
88     return nthl(tl(L), N - 1);
89 }
90
91 // 10.
92 list take(const list L, const int H) {
93     if (H == 0) return nil;
94     return cons(hd(L), take(tl(L), H - 1));
95 }
96
97 // 11.
98 list drop(const list L, const int B) {
99     if (B == 0) return L;
100    return drop(tl(L), B - 1);
101 }
102
103 // 12.
104 list sublist(const list L, const int H, const int B) {
105    return take(drop(L, B), H);
106 }
107
108 // 13.
109 list parban(const list L) {
110    if (L == nil) return nil;
111    if (tl(L) == nil) return nil;
112    if (hd(L) == hd(tl(L))) return cons(hd(L), parban(tl(L)));
113    return parban(tl(L));
114 }
115
116 // 13.*
117 list parbani(const list L, const list P) {
118    if (L == nil) return P;
119    if (tl(L) == nil) return P;
120    if (hd(L) == hd(tl(L))) return parbani(tl(L), cons(hd(L), P));
121    return parbani(tl(L), P);
122 }
123 list parbani(const list L) {
124    return parbani(L, nil);
125 }
126
127 // 14.
128 int inc(const int X) {
129    return X + 1;
130 }
131 list map(const fun1 F, const list L) {
132    if (L == nil) return nil;
133    return cons(F(hd(L)), map(F, tl(L)));
134 }
135 list lista_noveltje_2(const list L) {
136    return map(inc, L);
137 }
138
139 // 15.
140 int rdiv(const int A, const int B) {

```

```

141     return B / A;
142 }
143 int foldl(const fun2 F, const int A, const list L) {
144     if (L == nil) return A;
145     return foldl(F, F(hd(L), A), tl(L));
146 }
147
148 int plus1(const int A, const int B) {
149     return B + 1;
150 }
151 int length2(const list L) {
152     return foldl(plus1, 0, L);
153 }
154
155 // 16.
156 int first(const int B, const int J) {
157     return B;
158 }
159 int last2(const list L) {
160     return foldl(first, hd(L), L);
161 }
162
163 // 17.
164 int plus(const int A, const int B) {
165     return A + B;
166 }
167 int sum(const list L) {
168     return foldl(plus, 0, L);
169 }
170
171 // 18.
172 int sqplus(const int E, const int A) {
173     return E * E + A;
174 }
175 int sumsq(const list L) {
176     return foldl(sqplus, 0, L);
177 }
178
179 // 19.
180 int sq(const int X) {
181     return X * X;
182 }
183 int sumsq2(const list L) {
184     return sum(map(sq, L));
185 }
186
187 // 20.
188 // revapp
189 // mar Ceklában is van template, csak maskent kell elnevezni!
190 template <class Fun, class Elem>
191 Elem foldlt(const Fun F, const Elem A, const list L) {
192     if (L == nil) return A;
193     return foldlt(F, F(hd(L), A), tl(L));
194 }
195
196 // L megforditva A elott
197 list revapp(const list L, const list A) {
198     if (L == nil) return A;
199     return revapp(tl(L), cons(hd(L), A));
200 }
201
202 ///////////////////////////////////////////////////
203 // hasznos lista konstruktorok
204 list L(const int E1, const int E2) {
205     return cons(E1, cons(E2, nil));
206 }
207 list L(const int E1, const int E2, const int E3) {
208     return cons(E1, L(E2, E3));
209 }
210 list L(const int E1, const int E2, const int E3, const int E4) {

```

```

211     return cons(E1, L(E2, E3, E4));
212 }
213 list L(const int E1, const int E2, const int E3, const int E4,
214     const int E5) {
215     return cons(E1, L(E2, E3, E4, E5));
216 }
217
218 ///////////////////////////////////////////////////
219 // megoldasok tesztje
220
221 int main() {
222     //writeln(L3(10,20,30) == cons(10, cons(20, cons(30, nil))));
223     writeln(" 1. Csupa 0 es/vagy 1 szamjegy");
224     writeln(csupa01(100) == 1);
225     writeln(csupa01(2011) == 0);
226     writeln(" 2. Osztok szama");
227     writeln(osztok(12) == 6);
228     writeln(" 3. Legnagyobb kozos oszto -- naiv megoldas");
229     writeln(lnko(6, 12) == 6);
230     writeln(lnko(6, 11) == 1);
231     writeln(lnko(6, 10) == 2);
232     writeln(" 4. Legnagyobb kozos oszto -- euklideszi algoritmus");
233     writeln(lnko2(6, 12) == 6);
234     writeln(lnko2(6, 11) == 1);
235     writeln(lnko2(6, 10) == 2);
236     writeln(" 5. Lista hossza");
237     writeln(length(L(1,3,5)) == 3);
238     writeln(" 5.* Lista hossza - jobbrekurziv változat");
239     writeln(lengthi(L(1,3,5)) == 3);
240     writeln(" 6. Szamlista minden elemenek novelese");
241     writeln(lista_noveltje(L(1,5,2)) == L(2,6,3));
242     writeln(" 7. Lista utolso elemenek meghatarozasa");
243     writeln(last(L(5,1,2,8,7)) == 7);
244     writeln(" 8. Beszuras listaba adott helyre");
245     writeln(insert_nth(L(1,8,3,5), 2, 6) == L(1,8,6,3,5));
246     writeln(insert_nth(L(1,3,8,5), 3, 3) == L(1,3,8,3,5));
247     writeln(" 9. Lista adott sorszamu eleme");
248     writeln(nth(L(10,20,30), 3) == 30);
249     writeln(" 10. Lista adott hosszusagu prefixuma");
250     writeln(take(L(10,20,30,40,50), 3) == L(10,20,30));
251     writeln(" 11. Lista adott helyen kezdodo szuffixuma");
252     writeln(drop(L(10,20,30,40,50), 3) == L(40,50));
253     writeln(" 12. Reszlista kepzese");
254     writeln(sublist(L(10,20,30,40,50), 3, 1) == L(20,30,40));
255     writeln(" 13. Listaban parosaval elofordulo elemek listaja");
256     writeln(parban("aaabccabeedd") == "aaced");
257     writeln(" 13.* Listaban parosaval elofordulo elemek listaja - jobbrekurziv változat, ahol nem szamit az
eredmeny sorrendje");
258     writeln(parbani("aaabccabeedd") == "decaa");
259     writeln(" 14. A 6. feladat ujbolii megoldasa a map függvennyel");
260     writeln(lista_noveltje_2(L(1,5,2)) == lista_noveltje(L(1,5,2)));
261     writeln(" 15. Az 5. feladat ujbolii megoldasa a foldl függvennyel");
262     writeln(foldl(rdiv, 64, L(4,2,1)) == ((64/4)/2)/1); // vagyis 8.
263     writeln(length2(L(1,3,5)) == length(L(1,3,5)));
264     writeln(" 16. A 7. feladat ujbolii megoldasa a foldl függvennyel");
265     writeln(last2(L(5,1,2,8,7)) == last(L(5,1,2,8,7)));
266     writeln(" 17. Lista elemeinek osszege foldl függvennyel");
267     writeln(sum(L(10,20,30)) == 60);
268     writeln(" 18. Lista elemeinek negyzetosszege a foldl függvennyel");
269     writeln(sumsq2(L(10,20,30)) == 1400);
270     writeln(" 19. A 18. feladat megoldasa a map es sum függvenyekkel");
271     writeln(sumsq2(L(10,20,30)) == 1400);
272     writeln(" 20. ...melyikre emlekeztet a foldl függveny?");
273     const list L1 = "abc", L2 = "123";
274     writeln(foldlt(cons, L1, L2) == revapp(L2, L1));
275 }

```