```
/* -*- c++ -*-
 *  $Date: 2011-09-20 14:02:53 +0200 (k, 20 szept 2011) $
 *  Copyright (C) 2011, BME Declarative Programming Course
 *  2011.09.16. Gyakorlat - megoldasok
 */

#include "cekla.h"

////////////////////
// megoldasok

// 1.
int csupa01(const int N) {
    if (N == 0) return 1;
    if (N % 10 >= 2) return 0;
    return csupa01(N/10);
}

// 2.
// osztok(N,D) az N szam D-nel nem kisebb osztoinak szama
int osztok(const int N, const int D) {
    if (N < D) return 0;
    const int Ez = N % D == 0;
    return Ez + osztok(N, D+1);
}
int osztok(const int N) {
    return osztok(N, 1);
}

// 3.
// A es B legnagyobb olyan kozos osztoja, ami nem nagyobb D-nel
int lnko(const int A, const int B, const int D) {
//    if (A % D == 0)
//        if (B % D == 0)
//            return D;
    if (A % D + B % D == 0)
        return D;
    return lnko(A, B, D-1);
}
int lnko(const int A, const int B) {
    return lnko(A, B, A);
}

// 4.
int lnko2(const int A, const int B) {
    if (B == 0) return A;
    return lnko2(B, A % B);
}

// 5.
int length(const list L) {
    if (L == nil) return 0;
    return 1 + length(tl(L));
}

// 5.*
// listaHossza_i(L, H) az L lista hossza + H
int lengthi(const list L, const int H) {
    if (L == nil) return H;
    return lengthi(tl(L), H + 1);
}
int lengthi(const list L) {
    return lengthi(L, 0);
}

// 6.
list lista_noveltje(const list L) {
    if (L == nil) return nil;
    return cons(hd(L) + 1, lista_noveltje(tl(L)));
}

// 7.
// L nem lehet ures!
int last(const list L) {
    if (tl(L) == nil) return hd(L);
    return last(tl(L));
}

// 8.
list insert_nth(const list L, const int N, const int E) {
    if (N == 0) return cons(E, L);
    return cons(hd(L), insert_nth(tl(L), N - 1, E));
}

// 9.
int nth1(const list L, const int N) {
    if (N == 1) return hd(L);
    return nth1(tl(L), N - 1);
}

// 10.
list take(const list L, const int H) {
    if (H == 0) return nil;
    return cons(hd(L), take(tl(L), H - 1));
}

// 11.
list drop(const list L, const int B) {
    if (B == 0) return L;
    return drop(tl(L), B - 1);
}

// 12.
list sublist(const list L, const int H, const int B) {
    return take(drop(L, B), H);
}

// 13.
list parban(const list L) {
    if (L == nil) return nil;
    if (tl(L) == nil) return nil;
    if (hd(L) == hd(tl(L))) return cons(hd(L), parban(tl(L)));
    return parban(tl(L));
}

// 13.*
list parbani(const list L, const list P) {
    if (L == nil) return P;
    if (tl(L) == nil) return P;
    if (hd(L) == hd(tl(L))) return parbani(tl(L), cons(hd(L), P));
    return parbani(tl(L), P);
}
list parbani(const list L) {
    return parbani(L, nil);
}

// 14.
int inc(const int X) {
    return X + 1;
}
list map(const fun1 F, const list L) {
    if (L == nil) return nil;
    return cons(F(hd(L)), map(F, tl(L)));
}
list lista_noveltje_2(const list L) {
    return map(inc, L);
}

// 15.
int rdiv(const int A, const int B) {
```

```
    return B / A;
}
int foldl(const fun2 F, const int A, const list L) {
    if (L == nil) return A;
    return foldl(F, F(hd(L), A), tl(L));
}

int plus1(const int A, const int B) {
    return B + 1;
}
int length2(const list L) {
    return foldl(plus1, 0, L);
}

// 16.
int first(const int B, const int J) {
    return B;
}
int last2(const list L) {
    return foldl(first, hd(L), L);
}

// 17.
int plus(const int A, const int B) {
    return A + B;
}
int sum(const list L) {
    return foldl(plus, 0, L);
}

// 18.
int sqplus(const int E, const int A) {
    return E * E + A;
}
int sumsq(const list L) {
    return foldl(sqplus, 0, L);
}

// 19.
int sq(const int X) {
    return X * X;
}
int sumsq2(const list L) {
    return sum(map(sq, L));
}

// 20.
// revapp
// mar Ceklaban is van template, csak maskent kell elnevezni!
template <class Fun, class Elem>
Elem foldlt(const Fun F, const Elem A, const list L) {
    if (L == nil) return A;
    return foldlt(F, F(hd(L), A), tl(L));
}

// L megforditva A elott
list revapp(const list L, const list A) {
    if (L == nil) return A;
    return revapp(tl(L), cons(hd(L), A));
}

//////////////////
// hasznos lista konstruktorok
list L(const int E1, const int E2) {
    return cons(E1, cons(E2, nil));
}
list L(const int E1, const int E2, const int E3) {
    return cons(E1, L(E2,E3));
}
list L(const int E1, const int E2, const int E3, const int E4) {
    return cons(E1, L(E2,E3,E4));
}
list L(const int E1, const int E2, const int E3, const int E4,
       const int E5) {
    return cons(E1, L(E2,E3,E4,E5));
}

//////////////////
// megoldasok tesztje

int main() {
    //writeln(L3(10,20,30) == cons(10, cons(20, cons(30, nil))));
    writeln("  1. Csupa 0 es/vagy 1 szamjegy");
    writeln(csupa01(100) == 1);
    writeln(csupa01(2011) == 0);
    writeln("  2. Osztok szama");
    writeln(osztok(12) == 6);
    writeln("  3. Legnagyobb kozos oszto -- naiv megoldas");
    writeln(lnko(6, 12) == 6);
    writeln(lnko(6, 11) == 1);
    writeln(lnko(6, 10) == 2);
    writeln("  4. Legnagyobb kozos oszto -- euklideszi algoritmus");
    writeln(lnko2(6, 12) == 6);
    writeln(lnko2(6, 11) == 1);
    writeln(lnko2(6, 10) == 2);
    writeln("  5. Lista hossza");
    writeln(length(L(1,3,5)) == 3);
    writeln("  5.* Lista hossza - jobbrekurziv valtozat");
    writeln(lengthi(L(1,3,5)) == 3);
    writeln("  6. Szamlista minden elemenek novelese");
    writeln(lista_noveltje(L(1,5,2)) == L(2,6,3));
    writeln("  7. Lista utolso elemenek meghatarozasa");
    writeln(last(L(5,1,2,8,7)) == 7);
    writeln("  8. Beszuras listaba adott helyre");
    writeln(insert_nth(L(1,8,3,5), 2, 6) == L(1,8,6,3,5));
    writeln(insert_nth(L(1,3,8,5), 3, 3) == L(1,3,8,3,5));
    writeln("  9. Lista adott sorszamu eleme");
    writeln(nth1(L(10,20,30), 3) == 30);
    writeln("  10. Lista adott hosszusagu prefixuma");
    writeln(take(L(10,20,30,40,50), 3) == L(10,20,30));
    writeln("  11. Lista adott helyen kezdodo szuffixuma");
    writeln(drop(L(10,20,30,40,50), 3) == L(40,50));
    writeln("  12. Reszlista kepzese");
    writeln(sublist(L(10,20,30,40,50), 3, 1) == L(20,30,40));
    writeln("  13. Listaban parosaval elofordulo elemek listaja");
    writeln(parban("aaabccabeedd") == "aaced");
    writeln("  13.* Listaban parosaval elofordulo elemek listaja - jobbrekurziv valtozat, ahol nem szamit az eredmeny sorrendje");
    writeln(parbani("aaabccabeedd") == "decaa");
    writeln("  14. A 6. feladat ujboli megoldasa a map fuggvennyel");
    writeln(lista_noveltje_2(L(1,5,2)) == lista_noveltje(L(1,5,2)));
    writeln("  15. Az 5. feladat ujboli megoldasa a foldl fuggvennyel");
    writeln(foldl(rdiv, 64, L(4,2,1)) == ((64/4)/2)/1);  // vagyis 8.
    writeln(length2(L(1,3,5)) == length(L(1,3,5)));
    writeln("  16. A 7. feladat ujboli megoldasa a foldl fuggvennyel");
    writeln(last2(L(5,1,2,8,7)) == last(L(5,1,2,8,7)));
    writeln("  17. Lista elemeinek osszege foldl fuggvennyel");
    writeln(sum(L(10,20,30)) == 60);
    writeln("  18. Lista elemeinek negyzetosszege a foldl fuggvennyel");
    writeln(sumsq2(L(10,20,30)) == 1400);
    writeln("  19. A 18. feladat megoldasa a map es sum fuggvenyekkel");
    writeln(sumsq2(L(10,20,30)) == 1400);
    writeln("  20. ...melyikre emlekeztet a foldl fuggveny?");
    const list L1 = "abc", L2 = "123";
    writeln(foldlt(cons, L1, L2) == revapp(L2, L1));
}
```