Egy 100 férőhelyes busz 10 megállós útvonalon halad. A 0. és 10. megálló a két végállomás. A felszálló utasokat tömbben tároljuk, melynek minden eleme egyetlen utas adatait tartalmazó struktúra. A struktúra a felszállás és a leszállás megállójának sorszámát tartalmazza.
Írjon C függvényt, mely paraméterként veszi át a fent megadott tömböt, és egy másik, 10 egész értéket tároló tömböt is. A függvény a második tömbbe írja be az egyes megállókban leszálló utasok számát. A felszálló utasok az első tömbben megadott sorrend szerint szállnak fel, de csak annyian, ahányan az adott megállóban felférnek a buszra. Feltételezheti, hogy minden megállóban először leszállnak, majd felszállnak az utasok.

```
typedef struct {
  int fel, le;
} utas;

enum {F = 100, M = 10};

void busz(utas utasok[], int U, int le[])
{
  int letszam = 0, megallo;
  /* kimenet törlése */
  for (megallo = 0; megallo < M; ++megallo)
    le[megallo] = 0;
  /* minden megállóra */
  for (megallo = 0; megallo <= M; ++megallo) {
    int u;
    /* leszállás */
    if (megallo > 0)
      letszam -= le[megallo-1];
    /* felszállás */
    for (u = 0; u < U; ++u)
      if (utasok[u].fel == megallo && letszam < F) {
        letszam++;
        le[utasok[u].le-1]++;
      }
  }
}
```

Egy egyirányban láncolt, strázsa nélküli lista minden eleme egy rókát vagy egy nyulat tárol. Mikor leszáll az éjszaka, minden róka megeszi a rákövetkező nyulat. Ezután az életben maradt nyulak párzanak. Minden egymás melletti nyúlpár egy további nyulat hoz létre maguk között (a nyulak nemével most nem foglalkozunk). Amennyiben a lista 3 vagy több egymást követő nyulat tartalmaz, akkor a középső nyulak előre és hátrafelé is párzanak. A lista elején álló állatot hajnalban a vadász lelövi, és ezért töröljük a listából.
Adja meg a listaelem típusdefinícióját is, melyben nyulak és rókák között felsorolt típus (enum) használatával tesz különbséget! Írjon C függvényt, mely paraméterként egy láncolt listát vesz át, és módosítja azt egyetlen éjszaka szabályainak megfelelően. A függvény kezelje helyesen az üres és egyelemű listát is.
Például (dőlten szedve a kisnyulak):
R→N→N→R→R→N→N→N→R→N→N→N→N ⇒ N→R→R→N→*N*→N→R→N→*N*→N→*N*→N

```c
#include<stdlib.h>

typedef struct allat {
  enum {ROKA, NYUL} fajta;
  struct allat *kov;
} allat;

allat *ejszaka(allat* lista) {
  allat *elso, *hatso;
  if (lista == NULL)
    return NULL;
  hatso = lista;
  elso = hatso->kov;
  while (elso != NULL) {
    /* hami */
    if (hatso->fajta == ROKA && elso->fajta = NYUL) {
      elso = elso->kov; /* itt lehet, hogy NULL-ra lép! */
      free(hatso->kov);
      hatso->kov = elso;
    }
    /* szerelem */
    else if (hatso->fajta == NYUL && elso->fajta == NYUL) {
      hatso->kov = (allat*)malloc(sizeof(allat));
      hatso->kov->fajta = NYUL;
      hatso->kov->kov = elso;
    }
    /* léptetés */
    hatso = elso;
    if (elso != NULL) /* csak, ha lehet */
      elso = elso->kov;
  }
  /* vadász */
  hatso = lista;
  lista = lista->kov;
  free(hatso);
  return lista;
}
```

A FUTO.TXT szöveges fájl futóverseny eredményeit tartalmazza. A fájl minden sora egy versenyző rajtszámát és időeredményét tárolja. Minden tárolt érték előjel nélküli egész szám. A rajtszám értéke 1000 és 9999 közé esik, az ezres helyiérték a versenyző korosztályának sorszáma. Az időeredmény 0 és 7200 másodperc között változhat.
Írjon C programot, mely korcsoportonként kihirdeti, hogy hány másodperc időkülönbség volt az első és a 6. helyezett között az alábbi formában. Feltételezheti, hogy minden kategóriában legalább hatan célba értek.

```
1: 18 s
2: 42 s
   ...
9: 27 s
```

```c
#include <stdio.h>

enum {K = 9, H = 6}; /* K korcsoport, H helyezett */

int main(void)
{
  unsigned idok[K][H], rajt, ido, csop, hely;
  FILE *fp;
  /* inicializálás */
  for (csop = 0; csop < K; ++csop)
    for (hely = 0; hely < H; ++hely)
      idok[csop][hely] = 7201;
  /* olvasás fájl végéig */
  fp = fopen("FUTO.TXT", "r");
  while (fscanf(fp, "%u%u", &rajt, &ido) == 2) {
    csop = rajt/1000 - 1;
    /* hely megkeresése */
    for (hely = 0; hely < H && idok[csop][hely] < ido; hely++);
    /* beszúrás */
    if (hely < H) {
      unsigned j;
      for (j = H-1; j > hely; --j)
        idok[csop][j] = idok[csop][j-1];
      idok[csop][hely] = ido;
    }
  }
  fclose(fp);
  /* listázás */
  for (csop = 0; csop < K; ++csop)
    printf("%u: %u s\n", csop+1, idok[csop][H-1]-idok[csop][0]);
  return 0;
}
```