**A1. feladat**

A szókereső játékban egy kétdimenziós táblán szereplő betűkből kell szavakat előállítani. A játékos a táblán vízszintes és függőleges irányokban lépkedhet, lépéseinek sorozata egy szót ad ki. Definiáljon felsorolt típust az irányok tárolására. Írjon függvényt, mely paraméterként egy $30 \times 30$-as táblát, a kezdőpozíció sor- és oszlopszámát, valamint a játékos lépéseinek sorozatát tartalmazó tömböt kapja meg. A függvény adja vissza a kiadódó szót sztringként.

Feltételezheti, hogy a tábláról nem lép le a játékos, de a mezőket többször is érintheti.

```c
#include <stdlib.h>

typedef enum { B, J, F, L } irany;

char *kiolvas(char tabla[][30], int sor, int oszlop, irany iranyok[], int n)
{
  int i;
  char *szo = (char *)malloc((n+2)*sizeof(char));
  szo[0] = tabla[sor][oszlop];
  for (i = 0; i < n; ++i)
  {
    switch (iranyok[i])
    {
    case F: sor--; break;
    case L: sor++; break;
    case B: oszlop--; break;
    case J: oszlop++; break;
    }
    szo[i+1] = tabla[sor][oszlop];
  }
  szo[i+1] = '\0';
  return szo;
}
```

**A2. feladat**

Egy egyirányban láncolt kétstrázsás láncolt listában maffiózókat tartunk nyilván. Minden maffiózót az ereje (egész szám) jellemez. Egy új maffiózó megjelenésekor az első nála gyengébb maffiózó elé áll be a listába, majd szisztematikusan kiirtja a mögötte álló nála gyengébb kollégáit. Minden egyes leszámolás során saját ereje is csökken a legyőzött maffiózó erejével, de már nem változik meg a helye a listában. A leszámolást addig folytatja, amíg csak tudja, azaz amíg a közvetlen mögötte álló gyengébb nála.

Pl. Ha a lista a megjelenés előtt {X 12, 10, 8, 4, 3, 3, 1, X}, a megérkező maffiózó erőssége pedig 8, akkor a lista a maffiózó belépését követően {X, 12, 10, 8, **8**, ~~4~~, ~~3~~, 3, 1, X}, a leszámolások után pedig {X, 12, 10, 8, **1**, 3, 1, X}.

Definiáljon típust a maffiózók nyilvántartásához. Írjon függvényt, amely egy új maffiózó megjelenését modellezi. A függvény paraméterként az új maffiózó erősségét kapja meg, ez alapján dinamikusan hozza létre és illeszti a listába az új maffiózót, valamint gondoskodik a halott maffiózók eltüntetéséről is.

```c
#include <stdlib.h>

typedef struct maf {
  int ero;
  struct maf *kov;
} maffiozo;

void megerkezett(maffiozo *strazsa, int ero)
{
  maffiozo *p = strazsa, uj;
  while (p->kov->kov != NULL && p->kov->ero >= ero)
    p = p->kov;
  uj = (maffiozo *)malloc(sizeof(maffiozo));
  uj->ero = ero;
  uj->kov = p->kov;
  p->kov = uj;
  while (uj->kov->kov != NULL && uj->kov->ero < uj->ero)
  {
    uj->ero -= uj->kov-ero;
    p = uj->kov;
    uj->kov = p->kov;
    free(p);
  }
}
```

### A3. feladat

Írjon programot, mely a `kep.bin` bináris fájlban tárolt szürkeárnyalatos képről megállapítja és kiírja, hogy hány olyan sora van, melynek átlagos fényessége kisebb, mint a teljes kép átlagos fényessége. A fájl első két `unsigned` értéke a tárolt kép szélességét és magasságát adja meg, ezt követik a sorfolytonosan tárolt pixelek fényességei (0 és 255 között) `unsigned char` formátumban. A fájlt csak egyszer olvashatja végig. Egy sor átlagos fényességét külön függvényben számolja ki.

```c
#include <stdio.h>
#include <stdlib.h>

typedef unsigned char byte;

double av_brightness(byte *row, unsigned W)
{
  unsigned  i;
  double b = 0.0;
  for (i = 0; i < W; ++i)
    b += row[i]/(double)W;
  return b;
}

int main(void)
{
  unsigned W, H, h, result=0;
  byte *buffer;
  double B;

  FILE *f = fopen("kep.bin", "rb");
  fread(&W, sizeof W, 1, f);
  fread(&H, sizeof H, 1, f);
  buffer = (byte *)malloc(W*H*sizeof(byte));
  fread(buffer, sizeof(byte), W*H, f);
  fclose(f);

  B = av_brightness(buffer, W*H);
  for (h = 0; h < H; ++h)
    if (av_brightness(buffer+h*W, W) < B)
      result++;
  printf("%u", result);

  free(buffer);
  return 0;
}
```