Írjon egy `maxoszlop` nevű függvényt, ami paraméterként kap egy dinamikusan, soronként foglalt int mátrixot (a sorok kezdőcímeinek tömbjét), a sorok számát, és oszlopok számát; és visszaadja annak az oszlopnak az indexét, amelyikben az elemek összege a legnagyobb. Ha több maximális összegű oszlop is van, ezek közül a legnagyobb indexű oszlop indexét adja vissza.

Megjegyzés: A tesztekben található letrehoz, feltolt, és felszabadit függvényeket nem kell megírnia.

Szintaktikai hiba esetén a fordító által jelzett sor számából vonjon ki 40-et, hogy megkapja a szerkesztőben olvasható sorszámot!

For example:

Test	Result

`int sorok = 4, oszlopok = 5;`
`int **mtx = letrehoz(sorok, oszlopok);`
`feltolt(mtx, sorok, oszlopok, 0);`
`cout << "A legnagyobb osszegu oszlop indexe: " << maxoszlop(mtx, sorok, oszlopok) << "\n";`
`felszabadit(mtx, sorok);`
	`A legnagyobb osszegu oszlop i`

Question author's solution:

```
#include <climits>

int maxoszlop(int **mtx, int s, int o) {
    int maxsum = 0, maxi = 0;
    for (int i=0; i<o; i++) {
        int sum = 0;
        for(int j=0; j<s; j++) {
            sum += mtx[j][i];
        }
        if (sum > maxsum) {
            maxsum = sum;
            maxi = i;
        }
    }
    return maxi;
}
```

Adott a következő struktúra, amely országot, az ország Nobel-dijasainak számát és a lakosságot (millió főben) tartalmazza:

```
struct NobelDij{
 string country;
 unsigned int numWinnners;
 double millionPeople;
}
```

Írjon meg két függvényt:

- `void sortByRate(bool rate, NobelDij *nobels, int n)` - a rendezést valósítja meg, ha `rate` `true`, akkor Nobel díj / millió lakos, ha `false` akkor pedig a Nobel dij alapján csökkenő sorrendbe
- `void statistics(NobelDij *nobels, int n)` - írja ki orszagonként átlagosan hány millió lakos és átlagosan mennyi a Nobel-díjas szam

Szintaktikai hiba esetén a fordító által jelzett sor számából vonjon ki 40-at, hogy megkapja a szerkesztőben olvasható sorszámot!

For example:

Test	Result

`NobelDij nobels[] = {`
`{"AUS", 12, 24.4},`
`{"BEL", 10, 11.42},`
`{"CZE", 5, 10.6},`
`{"DNK", 13, 5.73},`
`{"FRA", 68, 64.97},`
`{"GRE", 2, 11.1},`
`{"HUN", 13, 9.7},`
`{"GER", 107, 82.11},`
`{"SPA", 8, 46.35},`
`{"ROM", 4, 19.67},`
`{"CHE", 26, 8.47},`
`{"SLN", 1, 2.},`
`{"SLK", 0, 5.45},`
`{"USA", 368, 324.45},`
`{"UKR", 2, 44.222},`
`};`
`sortByRate(true, nobels, 15);`
`printAll(nobels, 15);`
`statistics(nobels, 15);`
	`1. hely: CHE (3.07 Nobel dij / millio lakos) [Osszesen 26 Nobel]`
	`2. hely: DNK (2.27 Nobel dij / millio lakos) [Osszesen 13 Nobel]`
	`3. hely: HUN (1.34 Nobel dij / millio lakos) [Osszesen 13 Nobel]`
	`4. hely: GER (1.30 Nobel dij / millio lakos) [Osszesen 107 Nobel]`
	`5. hely: USA (1.13 Nobel dij / millio lakos) [Osszesen 368 Nobel]`
	`6. hely: FRA (1.05 Nobel dij / millio lakos) [Osszesen 68 Nobel]`
	`7. hely: BEL (0.88 Nobel dij / millio lakos) [Osszesen 10 Nobel]`
	`8. hely: SLN (0.50 Nobel dij / millio lakos) [Osszesen 1 Nobel]`
	`9. hely: AUS (0.49 Nobel dij / millio lakos) [Osszesen 12 Nobel]`
	`10. hely: CZE (0.47 Nobel dij / millio lakos) [Osszesen 5 Nobel]`
	`11. hely: ROM (0.20 Nobel dij / millio lakos) [Osszesen 4 Nobel]`
	`12. hely: GRE (0.18 Nobel dij / millio lakos) [Osszesen 2 Nobel]`
	`13. hely: SPA (0.17 Nobel dij / millio lakos) [Osszesen 8 Nobel]`
	`14. hely: UKR (0.05 Nobel dij / millio lakos) [Osszesen 2 Nobel]`
	`15. hely: SLK (0.00 Nobel dij / millio lakos) [Osszesen 0 Nobel]`
	`Orszagonkent atlagosan: 44.71 millio lakos`
	`Atlagos Nobel-dijas szam orszagonkent: 2.84`

Answer: (penalty regime: 0 %)

Reset answer

Question author's solution:

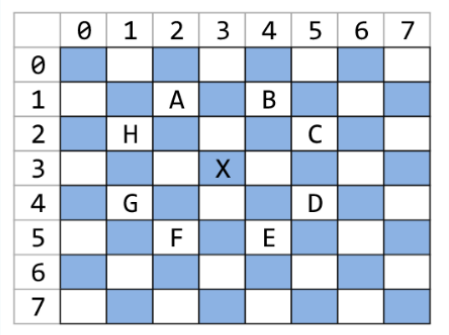
```
void sortByRate(bool rate, NobelDij *nobels, int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (rate){
                if (nobels[i].numWinnners / nobels[i].millionPeople > nobels[j].numWinnners / nobels[j].millionPeople)
                {
                    NobelDij t = nobels[i];
                    nobels[i] = nobels[j];
                    nobels[j] = t;
                }
            }
            else{
                if (nobels[i].numWinnners > nobels[j].numWinnners)
                {
                    NobelDij t = nobels[i];
                    nobels[i] = nobels[j];
                    nobels[j] = t;
                }
            }
        }
    }
}

void printAll(NobelDij *nobels, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << i+1 << ". hely: "<< nobels[i].country << " (" << fixed << setprecision(2) << nobels[i].numWinnners / nobels[i].millionPeople << " Nobel dij / millio lakos) [Osszesen " << nobels[i].numWinnners << " Nobel]\n";
    }
}

void statistics(NobelDij *nobels, int n)
{
    double meanN = 0.0, meanM = 0.0;
    for (int i = 0; i < n; i++)
    {
        meanN += nobels[i].numWinnners;
        meanM += nobels[i].millionPeople;
    }
    meanN /= n;
    cout << "Orszagonkent atlagosan: " << meanM / n << " millio lakos\n";
    cout << "Atlagos Nobel-dijas szam orszagonkent: " << meanN / n << "\n";
}
```

Írjon függvényt, ami egy 8x8-as sakktáblán fogadja el a ló lépéseit. A függvény neve legyen `lepesEllenor`. Az argumentunok: `bolS` jelölje a sort ahonnan, a `bolO` pedig az oszlopot ahonnan indul a lépés. A `baS` a sor, a `baO` pedig az oszlop ahova érkezik a ló. Térjen vissza `true`-val, ha szabályos a lépés `false`-al, minden más esetben (szabálytalan lépés, hibás input).

Segítség:



Szintaktikai hiba esetén a fordító által jelzett sor számából vonjon ki 10-et, hogy megkapja a szerkesztőben olvasható sorszámot!

For example:

Test	Result
`cout << (lepesEllenor(3,3,2,1) ? " helyes lepes" : " hibas lepes") << "\n";`	`[3,3] -> [2,1] helyes lepes`
`cout << (lepesEllenor(3,3,1,2) ? " helyes lepes" : " hibas lepes") << "\n";`	`[3,3] -> [1,2] helyes lepes`
`cout << (lepesEllenor(3,3,4,1) ? " helyes lepes" : " hibas lepes") << "\n";`	`[3,3] -> [4,1] helyes lepes`
`cout << (lepesEllenor(3,3,5,2) ? " helyes lepes" : " hibas lepes") << "\n";`	`[3,3] -> [5,2] helyes lepes`
`cout << (lepesEllenor(3,3,3,4) ? " helyes lepes" : " hibas lepes") << "\n";`	`[3,3] -> [3,4] hibas lepes`
`cout << (lepesEllenor(9,9,8,7) ? " helyes lepes" : " hibas lepes") << "\n";`	`[9,9] -> [8,7] helyes lepes`

Question author's solution:

```
bool lepesEllenor(unsigned short bolS, unsigned short bolO, unsigned short baS, unsigned short baO)
{
    cout << "["<< bolS << "," << bolO << "] -> [" << baS << "," << baO << "]";
    unsigned short sor = abs(bolS - baS);
    unsigned short oszlop = abs(bolO - baO);
    if((sor != 1 or oszlop != 2) and (oszlop != 1 or sor != 2)){
        return false;
    }
    else{
        return true;
    }
}
```

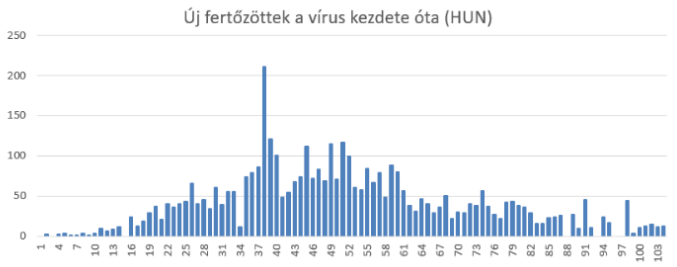
Készítsen függvényeket, amelyekkel fájlból olvas be egész számokat és azokat egy tömbbe illeszti. A számok a Magyarországon regisztrált COVID fertőzötteket jelentik a vírus megjelenésétől kezdve.

Valósítsa meg a 3 rövid függvényt:

- `void fajbolOlvas(string fajlnev, int adatok[], int db)` - Beolvassa *fajlnev*-ből az *adatok*-ba az *n* darabszámú adatot. A *fajlnev* nevű fájlnak legfeljebb az első *n* sorát, és az abban található adatokat helyezze el az adatok tömbben. Tesztesetként csak olyan eseteket adunk, ahol a fájl nagyobb vagy egyenlő *n*.
- `double atlag(const int adatok[], int db)` - Visszaadja a tömb átlag értékét.
- `double osszes(const int adatok[], int db)` - Visszaadja a tömb összértékét.

Például a `covid19.txt` tartalma:

```
2
0
2
3
1
1
3
1
3
9
6
8
11
0
23
12
....stb
```



Question author's solution:

```
// fájlból olvas
void fajbolOlvas(string fajlnev, int adatok[], int db)
{
    ifstream fajl(fajlnev.c_str());
    string adat;
    int i = 0;
    if (fajl.is_open())
    {
        while (getline(fajl, adat), !fajl.eof())
        {
            if (db > i)
            {
                adatok[i] = atoi(adat.c_str());
            }
            i++;
        }
    }
}

void kilistaz(const int adatok[], int db)
{
    for (int i = 0; i < db; i++)
    {
        cout << i << ". nap " << adatok[i] << " uj fertozott\n";
    }
}

double atlag(const int adatok[], int db)
{
    double atlag = 0.0;
    for (int i = 0; i < db; i++)
    {
        atlag += adatok[i];
    }
    return (atlag / db);
}

double osszes(const int adatok[], int db)
{
    double osszes = 0.0;
    for (int i = 0; i < db; i++)
    {
        osszes += adatok[i];
    }
    return osszes;
}
```

Készítsen `binTizesbe` függvényt, amely stringet vár bemenetként és decimális számot ad vissza. A bemeneti string egy bináris számsor, ennek megfelelő számot adjon vissza a függvény.

Szintaktikai hiba esetén a fordító által jelzett sor számából vonjon ki 10-et, hogy megkapja a szerkesztőben olvasható sorszámot!

For example:

Test	Result
`cout << binTizesbe("00000001") << "\n";`	`00000001 -- 1`
`cout << binTizesbe("00000010") << "\n";`	`00000010 -- 2`
`cout << binTizesbe("00000100") << "\n";`	`00000100 -- 4`
`cout << binTizesbe("00000101") << "\n";`	`00000101 -- 5`
`cout << binTizesbe("10000001") << "\n";`	`10000001 -- 129`

Question author's solution:

```
int binTizesbe(string n) {
  int szam = 0;
  cout << n << " -- ";
  for(unsigned i = 0; i < n.length(); i++) {
    szam = szam * 2 + n[i]-'0';
  }
  return szam;
}
```