Írjon egy olyan programot C++ nyelven, ami beolvas egy szót, és kiírja a szó Scrabble pontértékét.

A szó Scrabble pontértéke a benn lévő betűk pontszámainak összege. A betűk pontszámai a következők:

```
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q,  R, S, T, U, V, W, X, Y, Z
1, 3, 3, 2, 1, 4, 2, 4, 1, 8, 5, 1, 3, 1, 1, 3, 10, 1, 1, 1, 1, 4, 4, 8, 4, 10
```

Pl.: baba = 3 + 1 + 3 + 1 = 8

A program a standard bemenetről olvassa be a szót! A szó whitespace-eken kívül tetszőleges karaktereket tartalmazhat. A nem betű karakterek pontszáma 0, a kis- és nagybetűk között ne tegyen különbséget!

For example:

Input	Result
Baba	8
az	11
A-Z	11

Answer: (penalty regime: 0 %)

Reset answer

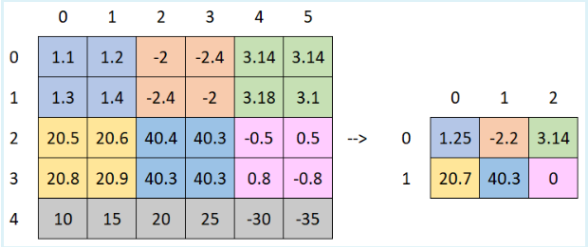
```
#include <iostream>
#include <string>

using namespace std;

int main() {
    int ertekek[26] = {1, 3, 3, 2, 1, 4, 2, 4, 1, 8, 5, 1, 3, 1, 1, 3, 10, 1, 1, 1, 1, 4, 4, 8, 4, 10};
    std::string szo;
    std::cin >> szo;
    int sum = 0;
    for (unsigned int i = 0; i < szo.length(); i++)
    {
        if (szo[i] <= 90 && szo[i] >= 65)
        {
            szo[i] = tolower(szo[i]);
        }
        if (szo[i] <= 122 && szo[i] >= 97)
        {
            sum += ertekek[szo[i] - 'a'];
        }
    }
    std::cout << sum;

}
```

Egy ***matrix*** struktúra az alábbi módon van definiálva:

```
struct matrix {
  int sorok; // A matrix sorainak szama
  int oszlopok; // A matrix oszlopainak szama
  double** m; // Matrix mutatotombos alakban megadva; 'sorok' darab 'double*' mutato, ami 'oszlop' elemu 'double' tombot cimez.
};
```

Készítsen egy ***kicsinyit*** nevű függvényt, ami paraméterben kap egy ilyen mátrix struktúrát, és visszaad egy másik, dinamikusan lefoglalt mátrixot, ami a paraméterben szereplőnek a "kicsinyített" mása:

- Feleannyi sora van, mint az eredetinek (lefelé kerekítve)
- Feleannyi oszlopa van, mint az eredetinek (lefelé kerekítve)
- Elemei az eredeti mátrix ugyanazon helyen lévő 4 szomszédos elemének átlaga
  - Lásd az alábbi ábrán
  - 1. sor 1. oszlopban az eredeti 1-2. sor 1-2. oszlopában lévő 4 érték átlaga
  - 1. sor 2. oszlopban az eredeti 1-2. sor 3-4. oszlopában lévő 4 érték átlaga
  - ...
  - 2. sor 2. oszlopban az eredeti 3-4. sor 3-4. oszlopában lévő 4 érték átlaga
  - ...
- *Ha az eredeti mátrix sorainak (vagy oszlopainak) száma páratlan, az utolsó sorát (oszlopát) hagyja figyelmen kívül*



A tesztkódokban található beolvasás és felszabadítás függvényeket nem kell elkészíteni. A mátrix dimenziói mindig nemnegatív számok, és a mátrix méretei ennek megfelelőek, ezek ellenőrzésével nem kell foglalkozni.

For example:

Test	Input	Result

matrix m = beolvas();
cout << "A beolvasott matrix:\n";
kiir(m);
matrix smaller = kicsinyit(m);
felszabadit(m);
cout << "\nKicsinyites utan:\n";
kiir(smaller);
felszabadit(smaller);
	5 6
	1.1 1.2 -2 -2.4 3.14 3.14
	1.3 1.4 -2.4 -2 3.18 3.1
	20.5 20.6 40.35 40.32 -0.5 0.5
	20.8 20.9 40.33 40.32 0.8 -0.8
	10 15 20 25 -30 -35
		A beolvasott matrix:
		1.1 1.2 -2 -2.4 3.14 3.14
		1.3 1.4 -2.4 -2 3.18 3.1
		20.5 20.6 40.35 40.32 -0.5 0.5
		20.8 20.9 40.33 40.32 0.8 -0.8
		10 15 20 25 -30 -35
		
		Kicsinyites utan:
		1.25 -2.2 3.14
		20.7 40.33 0

matrix m = beolvas();
cout << "A beolvasott matrix:\n";
kiir(m);
matrix smaller = kicsinyit(m);
felszabadit(m);
cout << "\nKicsinyites utan:\n";
kiir(smaller);
felszabadit(smaller);
	3 4
	1 2 3 4
	5 6 7 8
	9 10 11 12
		A beolvasott matrix:
		1 2 3 4
		5 6 7 8
		9 10 11 12
		
		Kicsinyites utan:
		3.5 5.5

Adott egy láncolt lista, melynek egy-egy elemét az alábbi, *adat* struktúra a forrásszöveg elején definiálja.

```
struct adat {
  int szam;
  adat* kov;
};
```

Implementálja a ***torolMin*** függvényt, ami paraméterben várja a lista első elemének címét (a horgonyt), kitörli a lista legkisebb számot tartalmazó elemét, majd visszaadja az így kapott list első elemének címét.

Ha csak 1 eleme volt a listának, akkor nullpointert adjon vissza. Ha a paraméter nullpointer, ne csináljon semii mást, csak adjon vissza nullpointert.

Ha a legkisebb érték többször szerepel a listában, *csak az első előfordulását törölje ki!*

For example:

Test	Result
`adat* horgony = torolMin(NULL);` `kiir(horgony);`	

`adat* horgony = NULL;`
`beszur(horgony, 3);`
`kiir(horgony);`
`horgony = torolMin(horgony);`
`kiir(horgony);`
`felszabadit(horgony);`
	`3`

`adat* horgony = NULL;`
`beszur(horgony, 3);`
`beszur(horgony, 3);`
`kiir(horgony);`
`horgony = torolMin(horgony);`
`kiir(horgony);`
`horgony = torolMin(horgony);`
`kiir(horgony);`
	`3 3`
	`3`

`adat* horgony = NULL;`
`beszur(horgony, 2);`
`beszur(horgony, -1);`
`beszur(horgony, 4);`
`beszur(horgony, 3);`
`kiir(horgony);`
`horgony = torolMin(horgony);`
`kiir(horgony);`
`horgony = torolMin(horgony);`
`kiir(horgony);`
`horgony = torolMin(horgony);`
`kiir(horgony);`
`felszabadit(horgony);`
	`3 4 -1 2`
	`3 4 2`
	`3 4`
	`4`

Answer: (penalty regime: 0 %)

Reset answer

Készítsen egy ***printUserStats*** függvényt, ami az 1. paraméterben megadott logfájlban megszámolja és kiírja, hogy a 2. paraméterben megadott nevű felhasználó **hány üzenetet küldött**, és **átlagosan hány karakter hosszúak voltak az üzenetei**. (A két számot két külön sorban írja ki, más ne szerepeljen a kimeneten!)

Az átlagos üzenethosszt dupla pontosságú lebegőpontos számként kezelje! Ha a felhasználónév nem található a fájlban, akkor az átlagos üzenethossz helyett írjon ki '-' karaktert. A példákban mindig létező fájl neve lesz megadva, nem kell kezelni a hiányzó fájlt.

A feldolgozandó logfájlok minden sora a következő formátumú: [időbélyeg] felhasználónév: üzenet

A felhasználónév csak alfanumerikus karakterekből állhat.

Példa fájl:

```
messenger.log:
[2020-06-17 13:09:16 UTC] Alice: hello
[2020-06-17 13:09:32 UTC] Bob: hi
[2020-06-17 13:09:45 UTC] Alice: what's up, Bob?
[2020-06-17 13:10:02 UTC] Bob: not much
[2020-06-17 13:10:02 UTC] cody: brb
```

For example:

Test	Result
`printUserStats("messenger.log", "Bob");`	`2` `5.5`
`printUserStats("messenger.log", "alice");`	`0` `-`
`printUserStats("messenger.log", "Alice");`	`2` `10`
`printUserStats("messenger.log", "cody");`	`1` `3`

Answer: (penalty regime: 0 %)

Az előző feladat folytatásaként, készítsen egy több felhasználóból álló statisztika lekérdezését megvalósító, ***queryUserStats*** függvényt.

Segítségképp meghívhatja a ***getUserStats*** függvényt, ami az előző feladatban szereplő függvényhez hasonlóan működik, de az 1 felhasználóra vonatkozó statisztikát nem kiírja, hanem visszaadja az alábbi struktúra példányaként:

```
struct UserStats {
    string user;
    int msgCount;
    double avgMsgLength;
};
```

A ***queryUserStats*** függvényben standard bemenetről olvassa be a lekérdezésben szereplő felhasználók számát, majd olvasson be ennyi felhasználónevet. Ezután írja ki a felhasználók statisztikáit az általuk küldött üzenetek száma szerint csökkenő, egyezés esetén ábécé sorrendben. A kimenet formátuma látható a példákban. A 0 üzenettel rendelkező felhasználók ne szerepeljenek a kimenetben!

For example:

Test	Input	Result

`queryUserStats("messenger.log");`	`2`
	`Alice`
	`Bob`
		`Alice 2 10`
		`Bob 2 5.5`

`queryUserStats("messenger.log");`	`4`
	`nObOdy`
	`cody`
	`Alice`
	`Bob`
		`Alice 2 10`
		`Bob 2 5.5`
		`cody 1 3`