1 kérdés

Hibás

1 pont szerezhető

A kérdés megjelölése

Készítsen struktúrát munkák ütemezésének számontartására. A munka struktúra 2 `string`et (munkát elvégző neve, munka azonosító) és két `int`et (munka kezdete, és munka hossza tartalmazzon). A kódból kiderül, de a tagok neve legyen: `elvegzo`; `azonosito`; `kezdes` és `hossz`.

Valósítsa meg a struktúradekalarációt és 2 rövid függvényt:

- `struct munka` - a deklaráció
- `bool utkozes(struct munka m1, struct munka m2)` - a munkák ütközésének figyelésére
- `void kiiratas(struct munka m)` - kiíratás *elvégző azonosító kezdés:befejezés* formátumban, a munkát elvégző neve szerint rendezve

Segítség

A legegyszerűbb logika a munkák ütközésére: ha az egyik munka **befejezése** nagyobb a másik munka kezdeténél **ÉS** a a másik munka **befejezése** nagyobb az egyik munka kezdeténél, akkor ütközik.

For example:

Test

```
struct munka munkak[DB] = {{"Cecil", "a5", 10 , 5}, {"Dani", "a4", 8 , 3}, {'
int i;
rendez(munkak, DB);
for(i = 0; i < DB; i++)
    kiiratas(munkak[i]);
```

```
#define DB 5

struct munka{
  string elvegzo;
  string azonosito;
  int kezdes;
  int hossz;
};

bool utkozes(struct munka m1, struct munka  m2){
    int bef1;
    bef1 = m1.kezdes + m1.hossz;
    int bef2;
    bef2 = m2.kezdes + m2.hossz;
    if(bef1 > m2.kezdes and bef2 > m1.kezdes) return true;
    else return false;
}

void kiiratas(struct munka m){
    cout << m.elvegzo << " " << m.azonosito << " "
    << m.kezdes << " " << m.kezdes << ':' << m.hossz << endl;
}

void rendez(struct munka m[], int n) {
    for(int i = n-1; i >= 1; i-- ) {
        for(int k = 0; k < i; k++) {
            if(m[k].elvegzo > m[k+1].elvegzo) {
                munka csere = m[k];
                m[k] = m[k+1];
                m[k+1] = csere;
            }
        }
    }
}
```

2 kérdés

Hibás

1 pont szerezhető

A kérdés megjelölése

Írjon egy string **szokozkiszed**(const string &s) és egy void **betubolszam**(string &s) függvényt.

A **szokozkiszed** az átadott stringből szedje ki a szóközöket, majd ezt a stringet adja vissza.

A **betubolszam** a paraméter stringben található kisbetűkből állítson elő számokat, mégpedig úgy, hogy '**a**'-ból, '**1**', '**b**'-ból '**2**', stb. legyen. Ezt a standard kimentre írja ki.

For example:

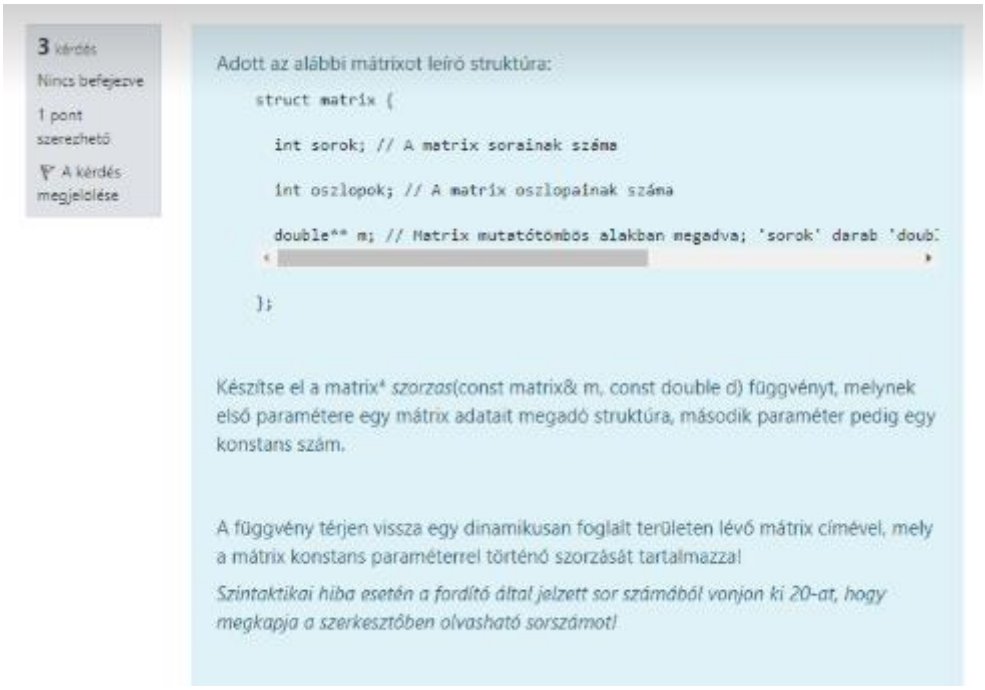
Test	Result

`string teszt1 = "a b c d e f"; cout << teszt1 << "\n";`
`cout << szokozkiszed(teszt1) << "\n";`
`betubolszam(teszt1);`
	`a b c d e f`
	`abcdef`
	`1 2 3 4 5 6`

```
#include <string>

string szokozkiszed(const string &s){
    int hosssz = s.length();
    string vissza;
    for(int i = 0; i < hosssz; i++){
        if (s[i] != ' ') {
            vissza += s[i];
        }
    }
    return vissza;
}

void betubolszam(const string &s){
    int hosssz = s.length();
    for(int i = 0; i < hosssz; i++){
        if(s[i] >= 'a' && s[i] <= 'i')
            cout << char(s[i] - 'a' + '1');
        else
            cout << s[i];
    }
}
```




```
matrix* szorzas(const matrix& m, const double d) {
  matrix* mtx = new matrix;
  mtx->sorok = m.sorok;
  mtx->oszlopok = m.oszlopok;
  mtx->m = new double*[mtx->sorok];
  for(int s=0; s<mtx->sorok; s++) {
    mtx->m[s] = new double[mtx->oszlopok];
    for(int o=0; o<mtx->oszlopok; o++) {
      mtx->m[s][o] = m.m[s][o]*d;
    }
  }
  return mtx;
}
```

4 kérdés

Nincs befejezve

1 pont szerezhető

A kérdés megjelölése

Írjon egy *minoszlop* 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 legkisebb. Ha több minimális összegű oszlop is van, ezek közül a legkisebb indexű oszlop indexét adja vissza.

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

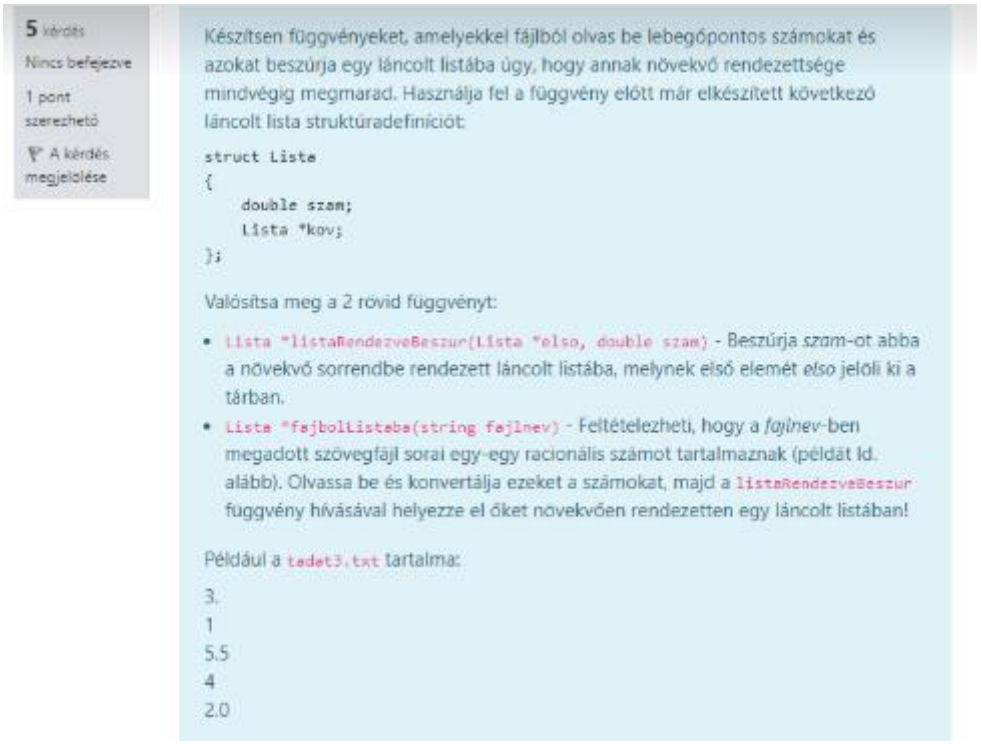
For example:

Test

```
int sorok = 4, oszlopok = 5;
int **mtx = letrehoz(sorok, oszlopok);
feltolt(mtx, sorok, oszlopok, 0);
cout << "A legkisebb osszegu oszlop indexe: " << minoszlop(mtx, sorok, oszlop
felszabadit(mtx, sorok);
```

```
#include <climits>

int minoszlop(int **mtx, int s, int o) {
    int minsum = INT_MAX, mini = -1;
    for (int i=0; i<o; i++) {
        int sum = 0;
        for(int j=0; j<s; j++) {
            sum += mtx[j][i];
        }
        if (sum < minsum) {
            minsum = sum;
            mini = i;
        }
    }
    return mini;
}
```



5 kérdés
Nincs befejezve
1 pont szerezhető
A kérdés megjelölése

Készítsen függvényeket, amelyekkel fájlból olvas be lebegőpontos számokat és azokat beszúrja egy láncolt listába úgy, hogy annak növekvő rendezettsége mindvégig megmarad. Használja fel a függvény előtt már elkészített következő láncolt lista struktúradefiníciót:

```
struct Lista
{
    double szam;
    Lista *kov;
};
```

Valósítsa meg a 2 rövid függvényt:

- `Lista *listaRendezveBeszur(Lista *elso, double szam)` - Beszúrja *szam*-ot abba a növekvő sorrendbe rendezett láncolt listába, melynek első elemét *elso* jelöli ki a tárban.
- `Lista *fajbolListaba(string fajlnev)` - Feltételezheti, hogy a *fajlnev*-ben megadott szövegfájl sorai egy-egy racionális számot tartalmaznak (példát ld. alább). Olvassa be és konvertálja ezeket a számokat, majd a `listaRendezveBeszur` függvény hívásával helyezze el őket növekvően rendezetten egy láncolt listában!

Például a `tadat3.txt` tartalma:

3.
1
5.5
4
2.0

```
struct Lista
{
    double szam;
    Lista *kov;
};

// rendezve szúr be
Lista *listaRendezveBeszur(Lista *elso, double szam)
{
    Lista *iter, *uj, *ideSzur = NULL;
    uj = new Lista;
    uj->szam = szam;
    // az elejére szúrjuk, ha a lista üres, vagy ez a legjobb eredmény
    if (elso == NULL || elso->szam > szam)
    {
        uj->kov = elso;
        elso = uj;
    }
    // nem az elejére beszúrás
    else
    {
        for (iter = elso; iter != NULL && iter->szam <= szam; iter = iter->kov)
            ideSzur = iter;
        uj->kov = iter;
        ideSzur->kov = uj;
    }
    return elso;
}


// fájlból olvas listába
Lista *fajbolListaba(string fajlnev)
{
    Lista *ures = NULL;
    ifstream fajl(fajlnev.c_str());
    string adat;
    if (fajl.is_open())
    {
        while (getline(fajl, adat), !fajl.eof())
        {
            ures = listaRendezveBeszur(ures, atof(adat.c_str()) );
        }
    }
    return ures;
}
```