**Programozás alapjai 1**

**Házi feladat dokumentáció**

**Papp Dominik Edvárd EAT3D9**

**Felhasználói dokumentáció**

**Éttermek a közelemben**

A program célja, hogy bárhol, bármikor, a helyzeti adataink és elvárásaink megadásával megtudjuk, hogy légvonalban egy adott távolságon belül mely éttermek felelnek meg a mi elvárásainknak.

**Forrásfájlok**

Egy szöveges fájl éttermek adatait tartalmazza egy sorban ’;’-vel elválasztva az alábbiak szerint:

Étterem egyedi azonosítója (egész szám)

Étterem neve (max. 50 karakter hosszú, szóközöket tartalmazhat)

Cím (max. 50 karakter hosszú, szóközöket tartalmazhat)

A konyha nemzetisége (max. 30 karakter hosszú)

GPS koordináták (Előbb az északi-szélességi majd a keleti-hosszúsági fokok)

Az étterem minősítése (két tizedes valós érték 1 és 5 között)

Árkategória ($ $$ $$$)

Teraszrész elérhetősége (Igen/ nem)

Egy másik szöveges fájl az éttermek asztalait és azoknak jelenlegi elérhetőségeit tartalmazza az alábbiak szerint.

Étterem egyedi azonosítója asztal típusa (hány fős) szabad aszlatok száma ’;’-vel elválasztva.

Pl.: 9867165;2;6

9867165;3;2

9867165;4;3

31857;6;1(másik étterem)

Egy harmadik szöveges fájl a felhasználó elvárásait és a saját Gps koordinátáit tartalmazza az első fájl sorrendje szerint (’;’-vel elválasztva a cím, név és az azonosító kihagyásával). Az étterem minősítését egy alsó határkent, az árkategóriát egy felső határként adja meg. A felhasználó megadhatja, hogy hányan szeretnének egy asztalnál ülni az étteremben, amennyiben ez mindegy, 0-t kell írni és így az első szabad asztal fog kiíródni)

A kör sugara (amelyen belül keresi az éttermet, valós szám m)

GPS koordináták (Előbb az északi-szélességi majd a keleti-hosszúsági fokok)

A konyha nemzetisége (több is felsorolható, max. 3)

Az étterem minősítése (két tizedes valós érték 1 és 5 között)

Árkategória ($ $$ $$$)

Teraszrész elérhetősége (Igen/ nem)

6(ennyi ember szeretne az étteremben egy asztalnál ülni)

Mindhárom fájlt adatbázisként kezeli a program, így elindítása előtt mindegyiket meg kell külön írni. Fontos továbbá ügyelni arra, hogy vagy minden legyen nagybetűvel, vagy minden legyen kisbetűvel írva. Amennyiben például az user.txt-ben a konyha kisbetűvel szerepel, az ettermek.txt-ben pedig nagybetűvel, akkor a program nem találja meg azt az éttermet és nem garantált a helyes működés.

**A program használata:**

A user.txt ettermek.txt és asztalok.txt fájlokat az alábbi példák szerint kell feltölteni. Az asztalok sorrendjének nem feltétlen kell megegyeznie az éttermek sorrendjével, a program el tudja dönteni melyik asztal melyik étteremhez tartozik.

Példák a fájlok egy-egy sorára:

user.txt: 4242;47.5062553054;19.0241112662;Magyar Amerikai;3.2;$$;Nem;5

ettermek.txt: 619355;Laposföld étterem;Jóllak utca 42/a;Amerikai;47.5066037598;19.0315785361;4.99;$$$;Nem

asztalok.txt: 619355;2;8

**Programozói dokumentáció**

**Forrásfájlok** (a zárójelek csak kommentek, nem kerülnek bele a fájlokba)

Az éttermek adatait éttermenként egy sorban tárolja az etterem.txt fájl. Egy soron beül az elemek a ’;’ karakterrel vannak elválasztva. Egy étteremre egy példa:

*123421 (étterem ID, unsigned int)*

*Laposföld étterem (neve: max. 50 karakter hosszú, szóközöket tartalmazhat)*

*Arany János utca 42/a (címe: max. 50 karakter hosszú, szóközöket tartalmazhat)*

*Olasz (a konyha nemzetisége: max. 30 karakter)*

*47.4983 19.0408 (gps koordinátái: Két valós érték. Előbb az északi-szélességi majd a keleti-hosszúsági fokok)*

*4.6 (minősítése: egy valós érték(5-1)*

*$$ (árkategória max. 3 karakter ($$$))*

*Igen (terasz elérhetősége: igen/nem)*

*„123421*;*Laposföld étterem*;*Arany János utca 42/a;Olasz*;*47.4983;19.0408*;*4.6*;*$$*;*Igen(\n ha nem EOF)”*

A fenti sor kerül az etterem.txt fájlba.

Az éttermek asztalait és azoknak elérhetőségeit egy külön fájl tárolja, az asztalok.txt fájl. A két fájl közötti kapcsolatot az étterem ID-ja teremti meg. Minden sorban szerepel az étterem ID-ja, majd, hogy hány fős asztalból, hány darab elérhető van. A nem elérhető asztalokat is feltünteti (feltesszük, hogy az elérhető asztalokból legalább egy a teraszon van, ha van terasz). A tagok itt is ’;’ karakterrel vannak elválasztva, de soronként vannak elválasztva a különböző asztalok. Egy étterem asztalaira egy példa.:

*123421;2;6*

*123421;3;2*

*123421;4;3*

*123421;6;1*

*123421;8;0*

A harmadik fájl a user elvárásait tartalmazza. Ennek a fájlnak a neve user.txt. Példa a user adataira:

*1024 (a kör sugara, amelyen belül keresi az éttermet) (egész szám méter egységben)*

*47.507350 19.026352(gps koordinátái: Két valós érték. Előbb az északi-szélességi majd a keleti-hosszúsági fokok)*

*Olasz Kínai Japán (A konyha nemzetisége. (max. 3 sorolható fel. Több mint egy megadása azt jelenti, hogy a usernek mindegy milyen az étterem konyhája, amíg azok közül az egyik))*

*4.1(Az étterem minősítése (alsó határ, legalább ennyi legyen))*

*$$(Árkategória, felső határ, legfeljebb ilyen drága legyen)*

*Igen (Teraszrész elérhetősége (teraszon akar ülni vagy nem))*

*6(A leülni kívánó személyek száma)*

*„1024; 47.507350;19.026352; Olasz Kínai Japán; 4.1;$$;Igen;6”*

**Adatszerkezet**

Az adatok tárolására egy két irányba láncolt fésűs lista van alkalmazva, melyben az első fájl adataiból képzett elemekből fog indulni a második fájl adataiból képzett láncolt lista, amely már csak előre láncolt. A lista dinamikusan foglalt elemekből áll. A listában az etterem elemek egymást minősítés szerinti csökkenő sorrendben követik, a gyorsabb működés érdekében.

Az első fájl adataiból képzett struktúra:(nagyhazi.c 5. sor)

typedef struct etterem{

    unsigned int id;

    char nev[50];

    char cim[50];

    char konyha[30];

    double eszaki;/\* északi keleti koordináták\*/

    double keleti;

    double minosites;/\*5.0-1.0\*/

    char arkat[4];/\*$ $$ $$$\*/

    char terasz[4];/\*igen/nem\*/

    struct etterem \*next;/\*következő étterem\*/

    struct etterem \*prev;/\*előző étterem\*/

    struct asztalok \*head;/\*asztalok listájára mutató pointer\*/

}etterem;

A második fájl adataiból képzett struktúra:(nagyhazi.c 20. sor)

typedef struct asztal{

    unsigned int id;/\*ez megegyezik az etterem.id -val\*/

    int ferohely;/\*asztal ülőhelyeinek száma\*/

    int szabad;/\*szabad asztalok száma\*/

    struct asztalok \*next;/\*az étterem asztalainak következő elemére mutat\*/

}asztal;

A harmadik fájlból képzett struktúra:(nagyhazi.c 27. sor)

typedef struct user{

    int sugar;

    double eszaki;/\* északi keleti koordináták\*/

    double keleti;

    char konyha1[20];

    char konyha2[20];/\*csak akkor kerül bele adat, ha több mint egy van megadva. Egyébként csupa '\0'\*/

    char konyha3[20];

    double minosites;

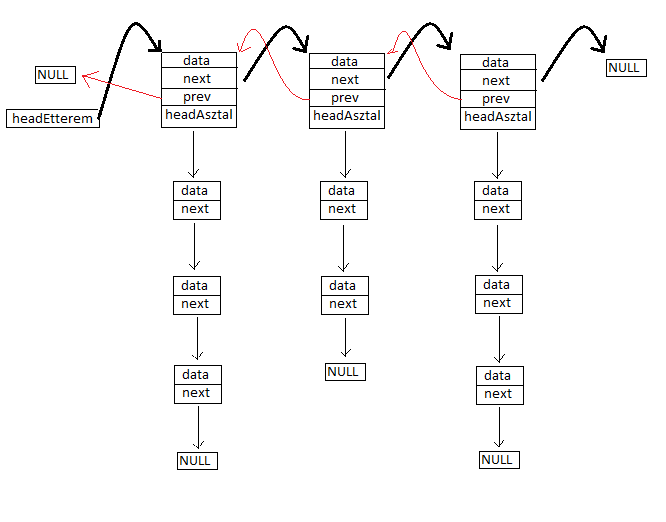
    char arkat[4];/\*$ $$ $$$ felsőhatár\*/

    char terasz[5];/\*igen/nem\*/

    int fo;/\*legalább ennyi fős asztal kell\*/

}user;

A listák viszonyai szemléltetve: (a data minden olyan struktúra elemet foglal magába, amely nem egy pointer)

****

**Adatok beolvasása és függvények**

A program átláthatósága érdekében segédfüggvények kerültek implementálásra.

etterem\* create\_etterem(void)

A create\_etterem() dinamikusan foglal „kinullázott” memóriát egy etterem struktúra elemnek.

Paraméterek: Nincs

Visszatérési érték: etterem\*

nagyhazi.c 40. sor

asztal\* create\_asztal(void)

A create\_asztal() dinamikusan foglal „kinullázott” memóriát egy asztal struktúra elemnek.

Paraméterek: Nincs

Visszatérési érték: asztal\*

nagyhazi.c 44.sor

user\* create\_user(void)

A create\_user() dinamikusan foglal „kinullázott” memóriát egy user struktúra elemnek.

Paraméterek: Nincs

Visszatérési érték: user\*

nagyhazi.c 48.sor

etterem\* insert\_etterem(etterem\* insert, etterem\* head)

Az insert\_etterem() beszúr a lácolt listába egy etterem elemet.

0 elem esetén a head a beszúrni kívánt elem lesz. Az elemet úgy szúrja be, hogy az éttermek a head->next irányába minősítés szerinti csökkenő sorrendben legyenek.

Paraméterek: etterem\* insert A beszúrni kívánt elem.

etterem\* head A láncolt lista kezdőcíme.

Visszatérési érték: etterem\*

nagyhazi.c 52. sor

void insert\_asztal(asztal\* insert, etterem\* head)

Az insert\_asztal() hozzáláncolja a megfelelő étteremhez tartozó asztalt az ahhoz tartozó asztal lista végéhez.

Paraméterek: asztal\* insert A beláncolandó asztal elem.

etterem\* head A láncolt lista kezdőcíme.

Visszatérési érték: Nincs

nagyhazi.c 90. sor

etterem\* beolvas\_etterem(etterem\* head)

A beolvas\_etterem() egy etterem elemet feltölt adatokkal.

A beolvas\_etterem() fő funkciója a fent említett, azonban a program egyik algoritmusa is. Hogy legyen hova beolvasni, létrehoz egy etterem elemet a create\_etterem() függvénnyel. Ezt az elemet tölti fel adatokkal, majd az insert\_etterem függvénnyel beilleszti a listába. A beolvas\_etterem() az ettermek.txt összes adatsorát feldolgozza és megépíti a fésűs lista gerincét. Kevés étteremhez tartozó adat esetén hibaüzenetet ír ki a standard outputra. Ilyenkor a program helyes működése nem garantált. A függvény kezeli az ettermek.txt fájl kinyitását és becsukását.

Paraméterek: etterem\* head A láncolt lista kezdőcíme.

Visszatérési érték: etterem\*

nagyhazi.c 107. sor

int beolvas\_asztal(etterem\* head)

A beolvas\_asztal() egy asztal elemet feltölt adatokkal.

Ennek a függvénynek a lefolyása hasonló a beolvas\_etterem()-éhez. Létrehoz a create\_asztal() függvénnyel egy elemet, ahova be tud olvasni, majd beilleszti a megfelelő helyre az insert\_asztal() függvénnyel. A beolvas\_asztal() függvény az asztal.txt összes adatsorát feldolgozza és nem megfelelő mennyiségű adat esetén hibaüzenetet ír ki a standard outputra. Ilyenkor a program megfelelő működése nem garantált. A függvény kezeli az asztalok.txt fájl kinyitását és becsukását.

Paraméterek: etterem\* head A láncolt lista kezdőcíme.

Visszatérési érték: int Normál működésnél 1, ha üres a user.txt akkor 0.

nagyhazi.c 125. sor

user\* beolvas\_user(void)

A beolvas\_user() egy user elemet feltölt adatokkal.

A függvény a create\_user() függvénnyel hozza létre a user elemet, ahova be fog olvasni. Nem megfelelő mennyiségű adat esetén hibaüzenetet ír ki a standard outputra. Ilyenkor a program megfelelő működése nem garantált. A függvény kezeli az user.txt fájl kinyitását és becsukását. Üres fájl esetén hibaüzenetet ír ki a standard outputra.

Paraméterek: Nincs

Visszatérési értek: user\*

nagyhazi.c 145. sor

etterem\* pop\_etterem(etterem\* pop, etterem\* head)

A pop\_etterem() felszabadít egy etterem elemet és az ahhoz tartozó teljes asztal listát.

A függvény külön kezel 4 esetet. Amikor a felszabadítani kívánt elem az egyetlen a listában, az első elem, az utolsó elem és amikor az elem tetszőleges helyen van a listában. Az első két esetben módosítja a lista kezdőcímére mutató pointert. A függvény újra láncolás után szabadít fel, így a sorrend továbbra is kihasználható. Felszabadítás előtt megvizsgálja, hogy tartozik-e asztal az étteremhez. Amennyiben nem, hibaüzenetet ír ki a standart outputra.

(fölösleges belőle etterem elemet készíteni ha le sem lehet ülni)

Paraméterek: etterem\* pop A felszabadítani kívánt elem.

etterem\* head A láncolt lista kezdőcíme.

Visszatérési érték: etterem\* Ha a felszabadítani kívánt elem az egyetlen a fésűs listában akkor a visszatérési értéke NULL. Ez kihasználható, ha a teljes listát kell felszabadítani.

nagyhazi.c 160. sor

etterem\* search\_for\_bad(user\* user, etterem\* head)

A search\_for\_bad() feladata, hogy javítsa a program futási idejét a fésűs lista minősítésbeli csökkenő sorrendjét kihasználva. A függvény megkeresi az első olyan elemet, amely már nem felel meg a user minősítésbeli elvárásainak és attól az elemtől kezdve felszabadítja a lista összes további elemét a pop\_etterem() többszöri meghívásával.

Paraméterek: user\* user A user adataiból álló struktúra.

etterem\* head A láncolt lista kezdőcíme.

Visszatérési érték: etterem\* Ha nincsen megfelelő étterem akkor NULL pointerrel tér vissza. Ez kihasználható a függvény meghívásakor.

nagyhazi.c 189. sor

double tavolsag(user\* user, etterem\* a)

A tavolsag() kiszámolja a user és egy étterem közötti távolságot.

Mivel földrajzi koordináták állnak rendelkezésre, ezért a függvény a Haversine formulát alkalmazza. A függvény csak közelítő értéket tud számolni, mivel a Föld nem szabályos gömb alakú. A függvény 6371km-es gömb (Föld) sugárral számol.

Paraméterek: user\* user A user adataiból álló struktúra.

etterem\* a Annak az étteremnek az adatait tartalmazó elem, amelynek a usertől mért távolságát kell kiszámolni.

Visszatérési érték: double Az étterem és a user közötti távolság méterben.

nagyhazi.c 202. sor

etterem\* compare(user\* user, etterem\* head)

A compare() a user minőségbeli elvárásán kívül az összessel összehasonlítja a fésűs listában található összes étteremet. Amennyiben az étterem nem felel meg az elvárásoknak a pop\_etterem() függvénnyel felszabadítja azt. Ez a függvény is a program egyik algoritmusa. Tovább ritkítja az elvárásoknak megfelelő listát. A függvény lefutása után már csak olyan elemek maradnak a listában, amelyek megfelelnek az elvárásoknak.

Paraméterek: user\* user A user adataiból álló struktúra.

etterem\* head A láncolt lista kezdőcíme.

Visszatérési érték: etterem\* Amennyiben nincsen megfelelő étterem akkor NULL pointerrel tér vissza. Ez kihasználható a függvény meghívásakor.

nagyhazi.c 211. sor

void print\_etterem(user\* user, etterem\* head)

A print\_etterem() kiírja egy étterem adatait a standard outputra.

A függvény rekurzív és egy algoritmus is. A függvény feldolgozza listában található első elemet és kiírja az étterem: nevét, címét, az első olyan asztal méretét, ahova le tud ülni a társaság és a usertől mért távolságot(tavolsag() függvénnyel). A kiírás után az elemet felszabadítja a pop\_etterem()-el és meghívja önmagát újra. A print\_etterem() kihasználja a pop\_etterem() azon tulajdonságát, hogy ha a lista üres, akkor a visszatérési érték NULL pointer. Ez a print\_etterem escape feltétele. A függvényből való távozás után a fésűs lista teljesen üres.

Paraméterek: user\* user A user adataiból álló struktúra.

etterem\* head A láncolt lista kezdőcíme.

Visszatérési érték: Nincs

nagyhazi.c 244. sor

**A main függvény**

A main függvény feladata a segédfüggvények kezelése. Létrehozza a fésűs listát beolvas\_etterem() és a beolvas\_asztal() függvények meghívásával (ebben a sorrendben!). Létrehozza a user struktúra elemet is, majd a search\_for\_bad() függvénnyel elkezdi csökkenteni a nem megfelelő éttermek elemeinek a számát. Ezután a compare() függvénnyel tovább csökkenti ezen elemek számát, majd így már csak a megfelelő elemek vannak a listában. Ilyenkor minden etterem elemet kiirat a standard outputra a print\_etterem() függvénnyel, ami felszabadítja az elemeket a kiírás után. Ezután a main függvény felszabadítja a user elemet és leáll a program. Amennyiben nincsen megfelelő étterem hibaüzenetet ír ki a standard outputra. Kezeli ha bármelyik fájl üres lenne.

nagyhazi.c 258. sor

**Algoritmusok**

A programnak 3 fő algoritmusa van melyeket a beolvas\_etterem(), compare() és print\_etterem() függvények valósítanak meg. Részletes leírás ezen algoritmusok működéséréről a függvények leírásainál található.

**Tesztelési dokumentáció**

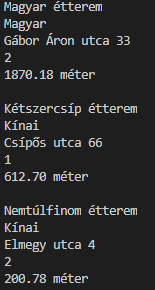
Input fájlok: user1.txt asztalok1.txt ettermek1.txt Nem felel meg egyik étterem sem az elvárásoknak.

Várt output: Magas elvárások hibaüzenet

Output: 

Input fájlok: user2.txt asztalok2.txt ettermek2.txt Van egy étterem amely megfelelő lenne, de nincsen ott megfelelő asztal.

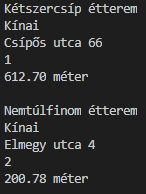
Várt output: Kétszercsíp étterem, Nemtúlfinom étterem, Magyar étterem (Egyélitt falatozóban nincs asztal)

Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input fájlok: user3.txt asztalok3.txt ettermek3.txt Van egy étterem amely megfelelő lenne, de túl drága.

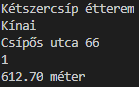
Várt output: Kétszercsíp étterem, Nemtúlfinom étterem (magyar étterem túl drága)

Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input fájlok: user4.txt asztalok4.txt ettermek4.txt Van egy étterem amely megfelelő lenne, de túl rossz minőségű.

Várt output: Kétszercsíp étterem (Nemúlfinom étterem túl rossz minőségű)

Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input fájlok: user5.txt asztalok4.txt ettermek4.txt Nincsen megfelelő minőségű étterem.

Várt output: Nincs elég jó étterem (egyik kínai sem elég jó minőségű már)

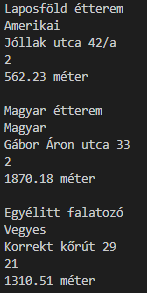
Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input fájlok: user6.txt asztalok6.txt ettermek6.txt Változott egy asztal elérhetőége.

Várt output: Magyar étterem, Laposföld étterem, Egyélitt falatozó (felszabadult egy 21 fős asztal az

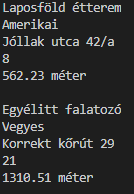
Egyélitt-ben)

Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input fájlok: user7.txt asztalok7.txt ettermek7.txt Van egy étterem amely megfelelő lenne, de nincsen ott megfelelő asztal.

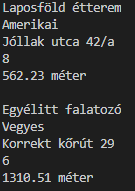
Várt output: Egyélitt falatozó, Laposföld étterem (az összes 6 vagy annál több fős asztal foglalt lett)

Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input fájlok: user7.txt asztalok8.txt ettermek7.txt Újranyílt egy asztal.

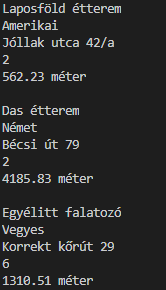
Várt output: Egyélitt falatozó, Laposföld étterem (lett egy 6 fős asztal)

Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Input fájlok: user8.txt asztalok8.txt ettermek7.txt Van egy teraszos étterem amely megfelelő lenne, de nincsen ott megfelelő asztal csak nagyobb.

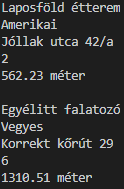
Várt output: teraszos éttermek a jó konyhával (jelen esetben az összes teraszos) (Das étteremnél nincs egy asztalos)

Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

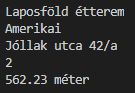
Input fájlok: user8.txt asztalok8.txt ettermek8.txt Van egy étterem amely megfelelő lenne, de nincsen ott terasz.

Várt output: teraszos éttermek a jó konyhával (jelen esetben az összes teraszos) (Das étteremnél nincs felújítják a teraszt)

Output: 

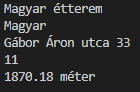
Input fájlok: user9.txt asztalok8.txt ettermek8.txt A user távolsági elvárásainak csak egy étterem felel meg.

Várt output: Laposföld étterem (a user elfáradt és nem akar olyan sokat sétálni)

Output: 

Input fájlok: user10.txt asztalok9.txt ettermek9.txt Csak egy étterem a listában, az jó.

Várt output: Az egy étterem a listában.

Output: 

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Üres ettermek.txt fájl

Várt output: Hibaüzenet.

Output: 