Házi Feladat

A feladat megoldását a Program.cs fájlba készítse el, melyet beadás előtt nevezzen át. A beadandó forrásfájl elnevezése a feladat azonosítója és a saját neptunkódja legyen alulvonással elválasztva, nagybetűkkel: **AZONOSÍTÓ_NEPTUNKOD.cs**

A feladattal kapcsolatos további információk az utolsó oldalon találhatók (ezen ismeretek hiányából adódó reklamációt nem fogadunk el!).

Készítsen szókereső alkalmazáshoz megoldó programot!

A szókereső játék célja, hogy a megadott táblázatban kell megkeresni a megadott szavakat (W), hogy megkapjuk a titkos üzenetet. A szavak a táblázatban szerepelhetnek vízszintesen, függőlegesen és átlósan (esetleg megfordítva), de mindig egyenes vonalban helyezkednek el. Minden szó csak egyszer szerepelhet a táblázatban, de az egyes betűk előfordulhatnak több szóban is (vagyis a szavak fedhetik egymást).

A megadott szavak megkeresését követőn lesznek olyan betűk melyek egyik szóban sem kerültek felhasználásra. Ezeket a betűket kell összegyűjteni balról jobbra, fentről lefelé, hogy megkapjuk a titkos szót.

Bemenet

- egyetlen fájl, aminek neve: `input.txt`
- a fájl felépítése:
  - *1. sor* - a megkeresendő szavak száma (N)
  - *következő N sor* - a táblázatban megkeresendő W szavak
  - $N+1$. *sor* - a táblázat sorainak (R) és oszlopainak (C) száma szóközzel elválasztva
  - *következő R sor* - táblázat soraiban található karakterek

Kimenet

- egyetlen fájl, aminek neve: `output.txt`
- a fájl csak a szabály alapján kiolvasott karakterek(et) tartalmazza

Megkötések

- $1 \leq N, R, C \leq 50$
- $1 \leq W$ karaktereinek száma ≤ 40
- $W \in \{ABCDEFGHIJKLMNOPQRSTUVWXYZ\}$
- minden W szó csak egyszer, egyféle képpen szerepel a táblázatban

Megjegyzés

- Amennyiben van rá lehetősége .NET Core helyett .NET Framework alkalmazás részeként készítse el a megoldást.

Példa

input.txt

```
2
BAL
BOB
3 3
BAL
BOB
BOR
```

output.txt

```
BOR
```

Értelmezés

A fájl első sora alapján kettő szót kell megkeresni a táblázatban: `BAL`, `BOB`. A fájl negyedik sora alapján a táblázat mérete 3×3, amiben a `BAL` az első sorban, a `BOB` a második sorban található. A harmadik sorban található karakterek érintetlenül maradtak, ezeket a szabály alapján (balról jobbra, fentről lefele) kiolvasva a `BOR` rejtett üzenetet kapjuk. Ezt az szöveget kell a fájlba írni.

Tesztesetek

Az alkalmazás helyes működését legalább az alábbi bemenetekkel tesztelje le!

input.txt	output.txt
2 BAL BOB 3 3 BAL BOB BOR	BOR
6 OLLO MAPPA ALMA CAT MOST POLC 5 4 MOST ALMT PLAA POLC AMLA	MA
14 TTL FMO UDD EDT UOT RD PFM PWP KUG ZK TTS YU SO HF 5 5 STTLK GONDZ UMODE KFPUR HPWPY	NOP

A fenti tesztesetek nem feltétlenül tartalmazzák az összes lehetséges állapotát a be- és kimenet(ek)nek, így saját tesztekkel is próbálja ki az alkalmazás helyes működését!

Tájékoztató

A feladattal kapcsolatosan általános szabályok:

- A feladat megoldásaként beadni vagy a betömörített solution mappa egészét vagy a Program.cs forrásfájlt kell (hogy pontosan melyiket, azt minden feladat külön definiálja), melynek elnevezése a feladat azonosítója és a saját neptunkódja legyen alulvonással elválasztva, nagybetűkkel: **AZONOSÍTÓ_NEPTUNKOD**[.zip|.cs]
- A megvalósítás során lehetőség szerint alkalmazza az előadáson és a laboron ismertetett programozási tételeket és egyéb algoritmusokat.
- Az alkalmazás elkészítése során mindenesetben törekedjen a megfelelő típusok használatára, illetve az igényes (*formázott, felesleges változóktól, utasításoktól mentes*) kód kialakítására, mely magába foglalja az elnevezésekkel kapcsolatos ajánlások betartását is (bővebben).
- A megoldásokhoz nem használhatók a beépített rendezőmetódusok (például: `Array.Sort`), a LINQ technológia (`System.Linq`), kivételkezelés (`try-catch-finally` blokk), a `goto`, a `continue` és a `break` (kivéve a `switch-case` szerkezetnél) utasítások, az alábbi gyűjtemények: `ArrayList`, `List`, `SortedList`, `Dictionary`, `Stack`, `Queue`, `Hastable`, a `var` az `object` és a `dynamic` kulcsszavak, illetve figyelembe kell venni a *Megkötések* pontban meghatározott további szabályokat.
- A leadott feladat megoldással kapcsolatos minimális elvárás a leírásban feltüntetett tesztesetek helyes futtatása, a *Megkötések* pontban definiáltaknak való megfelelés, ezeket leszámítva viszont legyen kreatív a feladat megoldásával kapcsolatban.
- A kiértékelés során csak a *Megkötések* pont szerenti helyes bemenettel lesz tesztelve az alkalmazás, a "tartományokon" kívüli értéket nem kell lekezelnie az alkalmazásnak.
- **Ne másoljon vagy adja be más megoldását!** Minden ilyen esetben az összes (felépítésben) azonos megoldás duplikátumként lesz megjelölve, melyek közül kizárólag, az időrendben elsőnek leadott lesz elfogadva.
- **Idő után leadott vagy helytelen elnevezésű megoldás vagy a kiírásnak nem megfelelő megoldás vagy fordítási hibát tartalmazó vagy (helyes bemenetet megadva) futásidejű hibával leálló kód nem értékelhető!**
- A feladat leírása az alábbiak szerint épül fel (* - opcionális):
  - *Feladat leírása* - a feladat megfogalmazása
  - **Bemenet* - a bemenettel kapcsolatos információk
  - **Kimenet* - az elvárt kimenettel kapcsolatos információk
  - *Megkötések* - a bemenettel, a kimenettel és az algoritmussal kapcsolatos megkötések, melyek figyelembevétele és betartása kötelező, továbbá az itt megfogalmazott bemeneti korlátoknak a tesztek minden eseteben eleget tesznek, így olyan esetekre nem kell felkészülni, amik itt nincsenek definiálva
  - **Megjegyzések* - további, a feladattal, vagy a megvalósítással kapcsolatos megjegyzések
  - *Példa* - egy példa a feladat megértéséhez
  - *Tesztesetek* - további tesztesetek az algoritmus helyes működésének teszteléséhez, mely nem feltétlenül tartalmazza az összes lehetséges állapotát a be- és kimenet(ek)nek
- **Minden eseteben pontosan azt írja ki és olvassa be az alkalmazás, amit a feladat megkövetel, mivel a megoldás kiértékelése automatikusan történik!** Így például, ha az alkalmazás azzal indul, hogy kiírja a konzolra a *"Kérem az első számot:"* üzenetet, akkor a kiértékelés sikertelen lesz, a megoldás hibásnak lesz megjelölve, ugyanis egy számot kellett volna beolvasni a kiírás helyett.
- A kiértékelés automatikusan történik, így különösen fontos a megfelelő alkalmazás elkészítése, ugyanis amennyiben nem a leírtaknak megfelelően készül el a megoldás úgy kiértékelése sikertelen lesz, a megoldás pedig hibás.
- Az automatikus kiértékelés négy részből áll:
  - Unit Test-ek - az alkalmazás futásidejű működésének vizsgálatára
  - Szintaktikai ellenőrzés - az alkalmazás felépítésének vizsgálatára
  - Duplikációk keresése - az azonos megoldások kiszűrésére
  - Metrikák meghatározása - tájékoztató jelleggel
- A kiértékelésnek eredményéből egy HTML report generálódik, melyet minden hallgató megismerhet.