



A feladat megoldása a Program.cs fájl legyen, melyet beadás előtt nevezzen át. A beadandó forrásfájl elnevezése a feladat azonosítója és a saját Neptun kódja 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 az S 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 követően lesznek olyan betűk melyek egyik szóban sem kerültek felhasználásra. A feladat ezeknek a karaktereknek az összegyűjtése balról jobbra, fentről lefelé haladva, így megkapva a titkos üzenetet, amelyet ki kell írunk a képernyőre.

Bemenet (Console)

- 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 (Console)

- a táblázatba a megmaradt karakterekből előállított S titkos üzenet

Megkötés(ek)

- $1 \leq N, R, C, |W| \leq 50$
- $W \in \{0 - 9a - zA - Z\}$
- minden W szó csak egyszer, egyféleképpen szerepel a táblázatban

Példa

Input	Output
2	BOR
BAL	
BOB	
3 3	
BAL	
BOB	
BOR	

Értelmezés

A konzolról elsőnek beolvasott érték alapján kettő szót kell megkeresni a táblázatban: BAL, BOB. A negyediknek beolvasott érték 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) kiolvastva a **BOR** rejtett üzenetet kapjuk, amit meg kell megjelenítenünk a kimeneten.



Tesztesetek

Az alkalmazás helyes működését legalább az alábbi bemenetekkel tesztelje le!

1.

Console input

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

Console output

```
BOR
```

2.

Console input

```
6
OLLO
MAPPA
ALMA
CAT
MOST
POLC
5 4
MOST
ALMT
PLAA
POLC
AMLA
```

Console output

```
MA
```

3.

Console input

```
14
TTL
FMO
UDD
EDT
UOT
RD
PFM
PWP
KUG
ZK
TTS
YU
SO
HF
5 5
STTLK
GONDZ
UMODE
KFPUR
HPWPY
```

Console output

```
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 megoldását egy Console App részeként kell elkészíteni, "Do not use top-level statements" beállítással, illetve a szükséges "using"-ok megtartása mellett.
- Megoldásként csak a Program.cs forrásfájl kell beadni, amelynek elnevezése a feladat azonosítója és a szerző saját Neptun kódja legyen alulvonással elválasztva, nagy betűkkel: **AZONOSÍTÓ_NEPTUNKOD.cs**
- A megvalósítás során lehetőség szerint alkalmazza az előadáson és a laboron ismertetett algoritmusokat, figyelembe véve a *Megkötések* pontban definiáltakat. Ezeket leszámítva viszont legyen kreatív a feladat megoldásával kapcsolatban.
- Az alkalmazás elkészítése során minden esetben törekedjen a megfelelő típusok használatára, illetve az igényes (*formázott, felesleges változóktól és utasításoktól mentes*) kód kialakítására, amely magába foglalja az elnevezésekkel kapcsolatos ajánlások betartását is (*bővebben*).
- **Ne másoljon vagy adja be más megoldását!** Minden ilyen esetben az összes (felépítésben) azonos megoldás duplikátumként lesz megjelölve és a megoldás el lesz utasítva.
- **A határidő 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 esetében 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 esetben 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 a számot:" üzenetet, akkor a kiértékelés sikertelen lesz, a megoldás hibásnak lesz megjelölve, ugyanis egy számot kellett volna beolvasni a kiírás helyett.
- A kiértékelés során csak a *Megkötések* pont szerinti helyes bemenettel lesz tesztelve az alkalmazás, a "tartományokon" kívüli értéket nem kell lekezelnie az alkalmazásnak.
- Elősegítve a fejlesztést, a beadott megoldás utolsó utasításaként szerepelhet egyetlen Console.ReadLine() metódushívás.
- Az automatikus kiértékelés négy részből áll:
 - Unit Test-ek - az alkalmazás futásidejű működésének vizsgálatára
 - Szintaktikai ellenőrzés - az alkalmazás felépítésének vizsgálatára
 - Tervezési irányelvek ellenőrzése - az alkalmazás "kinézetének" vizsgálatára
 - Duplikációk keresése - az azonos megoldások kiszűrésére
 - Metrikák meghatározása - tájékoztató jelleggel
- A kiértékelések eredményéből egy HTML report generálódik, melyet minden hallgató megismerhet.
- A feladat megoldásának minden esetben fordíthatónak és futtathatónak kell lennie **C# 10** és **.NET 6** keretrendszer használatával. Ettől függetlenül az elkészítés során használható egyéb változata a .NET keretrendszernek és a C# nyelvnek, azonban leadás előtt győződjön meg róla, hogy a megoldása kompatibilis a .NET 6 és C# 10 verzióval.



- A keretrendszer mellett további általános, nyelvi elemekkel való megkötés, melyek a házi feladatok során nem használhatók a megoldásban:
 - LINQ: `System.Linq` - *all query expressions within the namespace*
 - Attributes
 - Collections: `ArrayList`, `BitArray`, `DictionaryEntry`, `Hashtable`, `Queue`, `SortedList`, `Stack`
 - Keywords:
 - Modifiers: `abstract`, `async`, `event`, `external`, `in` - *generic modifier*, `new` - *member modifier*, `out` - *generic modifier*, `sealed`, `unsafe`, `virtual`, `volatile`
 - Statement: `break` - *in a loop*, `continue`, `goto`, `throw`, `try-catch-finally`, `checked`, `unchecked`, `fixed`, `lock`, `yield`
 - Namespace: `extern alias`
 - Generic type constraint: `new`, `where`
 - Access: `base`
 - Contextual: `add`, `partial` - *type, method*, `remove`, `required`, `when` - *filter condition*,
 - Query: `from`, `where`, `select`, `group`, `into`, `orderby`, `join`, `let`, `in`, `out`, `equals`, `by`, `ascending`, `descending`
 - Operators and Expressions:
 - Null-conditional operators: `?.` - *null conditional member access*, `[]` - *null conditional element access*
 - User-defined conversion operators: `implicit`, `explicit`
 - Pointer: `*` - *pointer*, `&` - *address-of*, `*` - *pointer indirection*, `->` - *pointer member access*, `[]` - *pointer element access*, `+`, `-`, `++`, `--` - *pointer arithmetic operators*
 - Assignment: `ref`
 - Lambda: `=>` - *expression, statement*
 - Others: `!` - *null forgiving*, `??` - *null coalescing*, `??=` - *null coalescing assignment*, `::` - *namespace alias qualifier*, `await`, `default` - *operator, literal*, `delegate`, `sizeof`, `stackalloc`, `with` - *expression, operator*
 - Types: `dynamic`, `interface`, `object`, `Object`, `var`, `struct`, `nullable`, `pointer`, `record`, `Tuple`, `Func<T>`, `Action<T>`, `Expression<T>`, `Nullable<T>`, `Span<T>`
 - Preprocessor directives: `#nullable`, `#if`, `#elif`, `#else`, `#endif`, `#define`, `#undefine`, `#undef`, `#error`, `#warning`, `#line`, `#pragma`
- Névterek, melyek kizárólagosan importálhatók a megoldásban (minden további névtér import törlésre kerül, illetve, ha az alábbiak közül valamelyik hiányzik, akkor hozzáadásra kerül a megoldáshoz):
 - `System`
 - `System.Collections.Generic`
 - `System.IO`
 - `System.Threading`