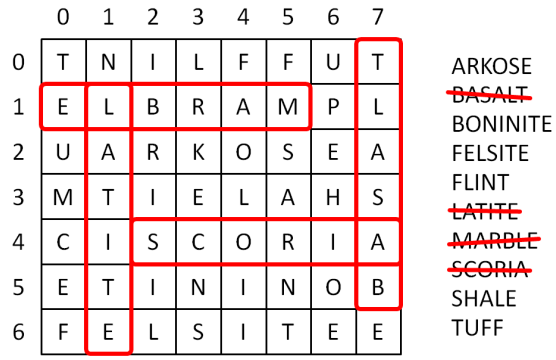
Word search

A word search puzzle is a word game that consists of the letters of words placed in a rectangular grid. The objective of this puzzle is to find and mark all the words hidden inside the grid. The words may be placed horizontally left-to-right or right-to-left, or vertically top-down or bottom-up. Often a list of the hidden words is provided, and many word search puzzles have a theme to which all the hidden words are related. The same letter may be used in multiple hidden words. When reading letters that are used in none of the hidden words from left to right and from top to bottom, a new word is formed that holds as the solution of the puzzle.



	0	1	2	3	4	5	6	7
0	T	N	I	L	F	F	U	T
1	E	L	B	R	A	M	P	L
2	U	A	R	K	O	S	E	A
3	M	T	I	E	L	A	H	S
4	C	I	S	C	O	R	I	A
5	E	T	I	N	I	N	O	B
6	F	E	L	S	I	T	E	E

ARKOSE
~~BASALT~~
BONINITE
FELSITE
FLINT
~~LATITE~~
~~MARBLE~~
~~SCORIA~~
SHALE
TUFF

Finish the word search puzzle and find the hidden word (PUMICE).

Assignment

Define a class WordSearch that supports at least the following methods:

1. An initialization method __init__ that takes three mandatory arguments: a number of rows r, a number of columns c and a string s that only contains letters. The initialization method must guarantee that each object of the class WordSearch has an attribute grid that represents a rectangular $r \times c$ grid which contains the given series of letters from left to right and from top to bottom. Take a look at the example given below, to see how the method should respond in case the given string s contains less than rc letters.
2. A method rows that takes no arguments and returns the number of rows of the word search grid.
3. A method columns that takes no arguments and returns the number of columns of the word search grid.
4. A method __str__ that takes no arguments and returns a string representation of the word search puzzle. The successive lines of this string representation correspond to the rows of the grid, where all letters on a single row are printed one after the other.
5. A method read that takes four mandatory arguments: *i*) a row index, *ii*) a column index, *iii*) a direction and *iv*) a length. Row and column indices start from zero. The method must return the word that is read from the grid in the given direction, starting at the cell in the given row and column. There are four possible directions in which words can be read from the grid, indicated by a pair of uppercase letters: LR (from left to right), RL (from right to left), TD (top-down), BU (bottom-up). The length of the word that must be read from the grid corresponds to the fourth argument passed to the method. The method must return an empty string in case no word can be read from the grid starting at the given position in the given direction (because reading runs against the borders of the grid).
6. A method search that takes a string argument representing a word that needs to be searched

in the grid of the word puzzle. The method must return in which direction and starting from which position the word can be read from the grid. The result must be returned as a tuple containing three elements: *i*) the row index of the starting position, *ii*) the column index of the starting position, and *iii*) the direction (as a string containing two uppercase letters). In searching the word in the grid, no distinction should be made between uppercase and lowercase letters. The method must return the value None in case the given word cannot be read from the grid in any of the four possible directions.

7. A method solution that returns the solution of the word search puzzle as an uppercase string. A list of hidden words must be passed as an argument to the method. The solution is found by reading the letters from the grid that are used in none of the hidden words from left to right and from top to bottom. Note that it is possible that the given list contains words that are not found in the grid. As some letters in the grid may be used in multiple hidden words, the best approach to find the solution is to replace the hidden words that are found in the grid by lowercase letters. After this is done for all hidden words, the solution may be formed from the remaining uppercase letters. For the sample word search puzzle given above, the following grid is obtained after having converted all hidden words into lowercase letters.

   tnilffut
   elbram**P**l
   Uarkosea
   Mt**I**elahs
   Ciscoria
   etininob
   felsite**E**

   From this grid, the solution of the word search puzzle reads as PUMICE.

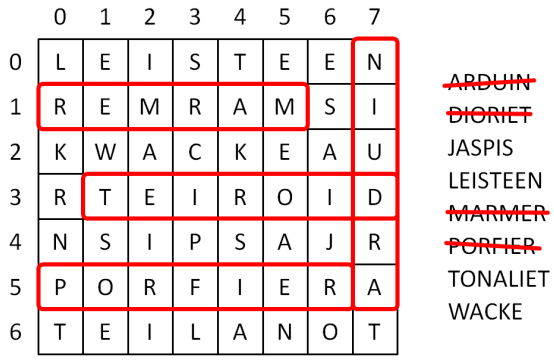
Example

```
>>> puzzle = WordSearch(7, 8, 'TNILFFUTELBRAMPLUARKOSEAMTIELAHSCISCORIAETININOBFELSITEE')
>>> print(puzzle)
TNILFFUT
ELBRAMPL
UARKOSEA
MTIELAHS
CISCORIA
ETININOB
FELSITEE
>>> puzzle.read(5, 3, 'BU', 6)
'NCEKRL'
>>> puzzle.read(3, 4, 'LR', 5)
''
>>> puzzle.search('marble')
(1, 5, 'RL')
>>> puzzle.search('chalk')
>>> puzzle.solution(['shale', 'basalt', 'felsite', 'boninite', 'latite', 'tuff', 'scoria', 'marble', 'arkose', 'flint'])
'PUMICE'

>>> puzzle = WordSearch(7, 8, 'BASALT')
Traceback (most recent call last):
AssertionError: too few letters given
```

Een woordzoeker is een puzzel waarbij men een aantal gegeven woorden moet zoeken in een rechthoekig rooster gevuld met letters. Deze woorden kunnen zowel van links naar rechts, van rechts naar links, van boven naar onder als van onder naar boven in het rooster neergeschreven

staan. Het is mogelijk dat dezelfde letter voor meerdere woorden moet gebruikt worden. De letters die voor geen enkel van de opgegeven woorden gebruikt worden, lezen van links naar rechts en van boven naar onder als een nieuw woord dat de oplossing van de puzzel vormt.



~~ARDUIN~~
~~DIORIET~~
JASPIS
LEISTEEN
~~MARMER~~
~~PORFIER~~
TONALIET
WACKE

Werk de woordzoeker verder af en vind het verborgen woord (SKARN).

Opgave

Definieer een klasse Woordzoeker die minstens ondersteuning biedt voor volgende methoden:

1. Een initialisatiemethode __init__ waaraan verplicht drie argumenten moeten doorgegeven worden: een aantal rijen r, een aantal kolommen k en een string s die enkel bestaat uit letters. Deze methode moet ervoor zorgen dat alle objecten van de klasse Woordzoeker een attribuut rooster hebben, dat een rechthoekig $r \times k$ rooster voorstelt waarin de opgegeven letters worden uitgespeld van links naar rechts en van boven naar onder. Alle letters moeten hierbij omgezet worden naar hoofdletters. Bestudeer onderstaand voorbeeld om te achterhalen wat er moet gebeuren indien de opgegeven string s minder dan rk letters bevat.
2. Een methode rijen zonder argumenten, die teruggeeft hoeveel rijen het rooster van de woordzoeker telt.
3. Een methode kolommen zonder argumenten, die teruggeeft hoeveel kolommen het rooster van de woordzoeker telt.
4. Een methode __str__ zonder argumenten die een stringvoorstelling van de woordzoeker teruggeeft. De opeenvolgende regels van deze stringvoorstelling bestaat uit de opeenvolgende rijen van het rooster, waarbij de letters op eenzelfde rij achter elkaar uitgeschreven worden.
5. Een methode lees waaraan verplicht de volgende vier argumenten moeten doorgegeven worden: *i*) een rijnummer, *ii*) een kolomnummer, *iii*) een richting en *iv*) een lengte. Voor rij- en kolomnummers begint men te tellen vanaf 0. De methode moet als resultaat het woord teruggeven dat in het rooster kan gelezen worden in de opgegeven richting beginnend vanaf de cel in de opgegeven rij en kolom. Er zijn vier mogelijke richtingen waarin het woord kan uitgelezen worden, die telkens worden aangegeven met een string van twee letters: LR (van links naar rechts), RL (van rechts naar links), BO (van boven naar onder) en OB (van onder naar boven). De lengte van het uit te lezen woord wordt aangegeven door het vierde argument. De methode moet de lege string teruggeven indien vanaf de opgegeven positie en in de opgegeven richting geen woord van de opgegeven lengte kan uitgelezen worden (omdat tegen de grenzen van het rooster wordt aangelopen).
6. Een methode zoek waaraan een woord als argument moet doorgegeven worden. De methode moet teruggeven in welke richting en vanaf welke positie in het rooster het woord kan uitgelezen worden. Dit resultaat wordt teruggegeven als een tuple met drie elementen:

i) het rijnummer van de startpositie, *ii*) het kolomnummer van de startpositie en *iii*) de richting (als een string bestaande uit twee letters). Bij het zoeken naar het opgegeven woord mag geen onderscheid gemaakt worden tussen hoofdletters en kleine letters. De methode moet de waarde None teruggeven indien het opgegeven woord in geen enkel van de vier mogelijke richtingen kan uitgelezen worden.

7. Een methode oplossing die de oplossing van de woordzoeker in hoofdletters als resultaat teruggeeft. Aan deze methode moet een lijst van woorden als argument doorgegeven worden. De oplossing bestaat dan uit de letters in het rooster die niet gebruikt worden door de opgelijste woorden, gelezen van links naar rechts en van boven naar onder. Merk op dat het best mogelijk is dat de lijst woorden bevat die niet in het rooster voorkomen. Aangezien sommige letters in het rooster door meerdere woorden kunnen gebruikt worden, kan je best op zoek gaan naar de oplossing door alle letters die gebruikt worden door woorden in het rooster te vervangen door de corresponderende kleine letter. De oplossing kan dan uitgelezen worden als de resterende hoofdletters. Voor het opgegeven voorbeeld ziet het rooster na omzetting van de opgegeven woorden naar kleine letters er dan bijvoorbeeld als volgt uit:

   leisteen
   remram**S**i
   Kwacke**A**u
   Rteiroid
   Nsipsajr
   porfiera
   teilanot

   Op basis van dit rooster kan de oplossing van de woordzoeker dan gelezen worden als SKARN.

Voorbeeld

```
>>> puzzel = Woordzoeker(7, 8, 'LEISTEENREMRAMSIKWACKEAURTEIROIDNSIPSAJRPORFIERATEILANOT')
>>> print(puzzel)
LEISTEEN
REMRAMSI
KWACKEAU
RTEIROID
NSIPSAJR
PORFIERA
TEILANOT
>>> puzzel.lees(5, 3, 'OB', 6)
'FPICRS'
>>> puzzel.lees(3, 4, 'LR', 5)
''
>>> puzzel.zoek('marmer')
(1, 5, 'RL')
>>> puzzel.zoek('bauxiet')
>>> puzzel.oplossing(['arduin', 'dioriet', 'jaspis', 'leisteen', 'marmer', 'porfier', 'tonaliet', 'wacke'])
'SKARN'

>>> puzzel = Woordzoeker(7, 8, 'LEISTEEN')
Traceback (most recent call last):
AssertionError: te weinig letters opgegeven
```