```
CS 519-005, Algorithms (MS/MEng-level), Winter 2018
HW10 - Challenge Problem - RNA Structure Prediction (6%)
This problem combines dynamic programming and priority queues.

Due Monday March 19, 11:59pm.
No late submission will be accepted.

Include in your submission: report.txt, rna.py.
Grading:
* report.txt -- 1%
* 1-best structure -- 2%
* number of structures -- 1%
* k-best structures -- 2%

Textbooks for References:
[1] KT Ch. 6.5 (DP over intervals -- RNA structure)
[2] KT slides: DP I (RNA section)
    http://www.cs.princeton.edu/~wayne/kleinberg-tardos/

***Please analyze time/space complexities for each problem in report.txt.
   [UPDATE] Also explain the algorithm of each question you solved.

1. Given an RNA sequence, such as ACAGU, we can predict its secondary structure
   by tagging each nucleotide as (, ., or ). Each matching pair of () must be
   AU, GC, or GU (or their mirror symmetries: UA, GC, UG).
   We also assume pairs can _not_ cross each other.
   The following are valid structures for ACAGU:

   ACAGU
   .....
   ...()
   ..(.)
   .(.).
   (...)
   ((.))

   We want to find the structure with the maximum number of matching pairs.
   In the above example, the last structure is optimal (2 pairs).

   >>> best("ACAGU")
   (2, '((.))')

   [UPDATE] Tie-breaking: arbitrary is fine. Don't worry as long as your structure
   is one of the correct best structures.

   [UPDATE] some other cases:

UUCAGGA
(3, '(((.)))')
GUUAGAGUCU
(4, '(.()((.)))')
GCACG
(2, '().()')
AUAACCUUAUAGGGCUCUG
(8, '.(((..)()()(((((((')
UUGGACUUGAGAAAAG
(5, '((...((()))...))')
UCAAUGGGUAGUAAAU
(6, '(((.)))((..(.)))')
UUUGGCACUUUCAGA
(6, '((((((.(..))))))')
ACACACCUUGUCCGUGAA
(6, '.((.(..)))(.)(.))')
GAUGCCGUGUAGUCCAAAGACUUCACCGUUGG
(14, '.()()(((((()))(((.((.)(.))(((((')
CGCGAAUAAAAAGGCACUGUU
(8, '()()((((....(().)))))')
ACGGCCAGUAAAGGUCAUAUACGCGGAAUGACAGGUCUAUCUAC
(19, '.()((((.)(..))(((.()()(((((((.)(((((((((())')
```

```
UGGGUGAGUCGCACACUCUGCGUACUCUUUCCGUAAUU
(15, '.(((((((((.()).(.))))(((()))).((...))))))')
AUACGUCGGGGACAAGAAUUACGG
(8, '.(.(((((((..(..)..)))))))')
AGGCAUCAAACCCUGCAUGGGAGCACCGCCACUGGCGAUUUUGGUA
(20, '.(()()).((((((()))))(((().()((((.))))()))))))()')
CGAGGUGGCACUGACCAAACACCACCGAAAC
(9, '.(.((((((.)((.)..))).).)...()...)')
CGCCGUCCGGGCGCGCCUUUUACGUAGAUUU
(12, '.(..(...((((((((((((.((((((((((')
CAUCGGGGUCUGAGAUGGCCAUGAAGGGCACGUACUGUUU
(18, '(()())((((((.))))(((((()))(.(.().()))))))))')
AACCGCUGUGUCAAGCCCAUCCUGCCUUGUU
(11, '(((.(..(.((.)(((...().))()))))))))')
```

2. Total number of all possible structures

```
>>> total("ACAGU")
6
```

3. k-best structures: output the 1-best, 2nd-best, ... kth-best structures.

```
>>> kbest("ACAGU", 3)
[(2, '((.))'), (1, '(...)'), (1, '.(.).')]
```

The list must be sorted.
[UPDATE] Arbitray tie-breaking is fine.

In case the input k is bigger than the number of possible structures, output all.

Sanity check: kbest(s, 1)[0][0] == best(s)[0] for each RNA sequence s.

All three functions should be in one file: rna.py.

See testcases at the end (also in test.txt on canvas).

Debriefing (required!): --------------------------

0. What's your name?
1. Approximately how many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Did you work on it mostly alone, or mostly with other people?
4. How deeply do you feel you understand the material it covers (0%–100%)?
5. Which part(s) of the course you like the most so far?
6. Which part(s) of the course you dislike the most so far?

This section is intended to help us calibrate the homework assignments.
Your answers to this section will *not* affect your grade; however, skipping it
will certainly do.

TESTCASES:

for each sequence s, we list three lines:
best(s)
total(s)
kbest(s, 10)

```
ACAGU
(2, '((.))')
6
[(2, '((.))'), (1, '.(.).'), (1, '..(.)'), (1, '...()'), (1, '(...)'), (0, '.....')]
------
AC
(0, '..')
1
[(0, '..')]
```

```
------
GUAC
(2, '(())')
5
[(2, '(())'), (1, '()..'), (1, '.().'), (1, '(..)'), (0, '....')]
------
GCACG
(2, '().()')
6
[(2, '().()'), (1, '(..).'), (1, '()...'), (1, '.(..)'), (1, '...()'), (0, '.....')]
------
CCGG
(2, '(())')
6
[(2, '(())'), (1, '(.).'), (1, '.().'), (1, '.(.)'), (1, '(..)'), (0, '....')]
------
CCCGGG
(3, '((()))')
20
[(3, '((()))'), (2, '((.)).'), (2, '(.()).'), (2, '.(()).'), (2, '.(().)'), (2, '.((.))'),
(2, '((.).)'), (2, '(.(.))'), (2, '(.().)'), (2, '((..))')]
------
UUCAGGA
(3, '(((.)))')
24
[(3, '(((.)))'), (2, '((.).).'), (2, '((..)).'), (2, '(.(.)).'), (2, '((.))..'), (2, '.
((.)).'), (2, '.((.).)'), (2, '.((..))'), (2, '((..).)'), (2, '((.)..)')]
------
AUAACCUA
(2, '.((...))')
19
[(2, '((.)..).'), (2, '()...).'), (2, '()(...).'), (2, '().(..).'), (2, '()....()'), (2,
'.()(..).'), (2, '.()...()'), (2, '.(.)..()'), (2, '.((...))'), (2, '.(.(..))')]
------
UUGGACUUG
(4, '(()((.)))')
129
[(4, '(()(.)()'), (4, '(()((.)))'), (3, '(().)..()'), (3, '(().).(.)'), (3, '(().)(..)'),
(3, '((.))..()'), (3, '((.)).(.)'), (3, '((.))(..)'), (3, '(())(..).'), (3, '(())(.)..')]
------
UUUGGCACUA
(4, '(.()()(.))')
179
[(4, '((()).).()'), (4, '((.())).()'), (4, '(.())).()'), (4, '.(()()).()'), (4, '.(()()
(.))'), (4, '((()).(.))'), (4, '((.)()(.))'), (4, '(()())..)'), (4, '(.()()(.))'), (3,
'((()).)...')]
------
GAUGCCGUGUAGUCCAAAGACUUC
(11, '(((((()((((.))))((.))))')
2977987
[(11, '((()))((((().))))((.))))'), (11, '((()))((((()()).)(((.))))'), (11, '((()))((((()
(.)))))((.))))'), (11, '((()()((((()(.)))))((.))))'), (11, '(((((()((((()).))))(.))))'), (11,
'(((((()((((.))))((.))))'), (11, '((()()((((()()).)(((.))))'), (11, '((()()((((()()).))
(((.))))'), (11, '(((((()((((()()).))((.))))'), (10, '((()()((((()().).)((.))).')]
------
AGGCAUCAAACCCUGCAUGGGAGCG
(10, '.(()()).((((()())))).())')
560580
[(10, '.(()()).((((()()))).)()'), (10, '.(()()).((((()())))).)()'), (10, '.(()()).((((()()))).)()'), (10, '.(()()).((((()()))).())'), (10, '.(()()).((((()()))).())'),
(10, '.(()()).((((()()))).())'), (9, '((.).)(...(.(()()))).)()'), (9, '((.).)(...(((.)
(()))).)()'), (9, '((.).)(...(.(()()))).)()'), (9, '((.).)(...((.()()))).)()')]
------
```