```
CS 519-005, Algorithms (MS/MEng-level), Winter 2018
HW6 - DP (part 2)

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

Need to submit: report.txt, knapsack_unbounded.py, knapsack_bounded.py.
knapsack_bounded.py will be graded for correctness (1%).

Textbooks for References:
[1] CLRS Ch. 15
[2] KT Ch. 6, freely available online (strongly recommended!):
    http://www.aw-bc.com/info/kleinberg/assets/downloads/ch6.pdf
[3] Wikipedia: Knapsack (unbounded and 0/1)
[4] Wikipedia: Longest Increasing Subsequence

Please answer time/space complexities for each problem in report.txt.

0. For each of the coding problems below:
   (a) Describe an exhaustive solution, and analyze compleixty (might be exponential).
   (b) Describe a greedy solution, and analyze complexity.
   (c) Show a counterexample to the greedy solution.
   (d) Theoretically, is the top-down solution faster, or the bottom-one one faster?
   (e) Empirically, which one is faster? (Try some long random lists)

1. Unbounded Knapsack

   You have n items, each with weight w_i and value v_i, and has infinite copies.
   **All numbers are positive integers.**
   What's the best value for a bag of W?

   >>> best(3, [(2, 4), (3, 5)])
   (5, [0, 1])

   the input to the best() function is W and a list of pairs (w_i, v_i).
   this output means to take 0 copies of item 1 and 1 copy of item 2.

   tie-breaking: *reverse* lexicographical: i.e., [1, 0] is better than [0, 1]:
   (i.e., take as much from item 1 as possible, etc.)

   >>> best(3, [(1, 5), (1, 5)])
   (15, [3, 0])

   >>> best(3, [(1, 2), (1, 5)])
   (15, [0, 3])

   >>> best(3, [(1, 2), (2, 5)])
   (7, [1, 1])

   >>> best(58, [(5, 9), (9, 18), (6, 12)])
   (114, [2, 4, 2])

   >>> best(92, [(8, 9), (9, 10), (10, 12), (5, 6)])
   (109, [1, 1, 7, 1])

   Q: What are the time and space complexities?

   filename: knapsack_unbounded.py

2. [WILL BE GRADED]
   Bounded Knapsack

   You have n items, each with weight w_i and value v_i, and has c_i copies.
   **All numbers are positive integers.**
   What's the best value for a bag of W?

   >>> best(3, [(2, 4, 2), (3, 5, 3)])
   (5, [0, 1])
   the input to the best() function is W and a list of triples (w_i, v_i, c_i).

   tie-breaking: same as in p1:

   >>> best(3, [(1, 5, 2), (1, 5, 3)])
   (15, [2, 1])

   >>> best(3, [(1, 5, 1), (1, 5, 3)])
   (15, [1, 2])

   >>> best(20, [(1, 10, 6), (3, 15, 4), (2, 10, 3)])
   (130, [6, 4, 1])

   >>> best(92, [(1, 6, 6), (6, 15, 7), (8, 9, 8), (2, 4, 7), (2, 20, 2)])
   (236, [6, 7, 3, 7, 2])

   Q: What are the time and space complexities?

   filename: knapsack_bounded.py

3. Longest (Strictly) Increasing Subsequence

   input/output are lower-case strings:

   >>> lis("aebbcg")
   "abcg"

   >>> lis("zyx")
   "z"

   tiebreaking: arbitrary. any optimal solution is ok.

   Q: What are the time and space complexities?

   filename: lis.py


PLEASE COME UP WITH MORE TESTCASES FOR EACH PROBLEM!
THESE EXISTING CASES ARE WAY TOO TRIVIAL.


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.
```