```python
# analysis.py
# -----------
# Licensing Information:  You are free to use or extend these projects for
# educational purposes provided that (1) you do not distribute or publish
# solutions, (2) you retain this notice, and (3) you provide clear
# attribution to UC Berkeley, including a link to http://ai.berkeley.edu.
#
# Attribution Information: The Pacman AI projects were developed at UC Berkeley.
# The core projects and autograders were primarily created by John DeNero
# (denero@cs.berkeley.edu) and Dan Klein (klein@cs.berkeley.edu).
# Student side autograding was added by Brad Miller, Nick Hay, and
# Pieter Abbeel (pabbeel@cs.berkeley.edu).


######################
# ANALYSIS QUESTIONS #
######################

# Set the given parameters to obtain the specified policies through
# value iteration.

def question2a():
    """
      Prefer the close exit (+1), risking the cliff (-10).
    """
    answerDiscount = 0.5
    answerNoise = 0.1
    answerLivingReward = -5
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'

def question2b():
    """
      Prefer the close exit (+1), but avoiding the cliff (-10).
    """
    answerDiscount = 0.5
    answerNoise = 0.1
    answerLivingReward = -1
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'

def question2c():
    """
      Prefer the distant exit (+10), risking the cliff (-10).
    """
    answerDiscount = 0.7
    answerNoise = 0.1
    answerLivingReward = 0
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'

def question2d():
    """
      Prefer the distant exit (+10), avoiding the cliff (-10).
    """
    answerDiscount = 0.9
    answerNoise = 0.2
    answerLivingReward = 0
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'

def question2e():
    """
      Avoid both exits and the cliff (so an episode should never terminate).
    """
    answerDiscount = 1
    answerNoise = 0.1
    answerLivingReward = 100
    return answerDiscount, answerNoise, answerLivingReward
    # If not possible, return 'NOT POSSIBLE'

if __name__ == '__main__':
    print('Answers to analysis questions:')
    import analysis
    for q in [q for q in dir(analysis) if q.startswith('question')]:
        response = getattr(analysis, q)()
        print('  Question %s:\t%s' % (q, str(response)))
```