MCS 253P - Lab 9

***No need to display case testing in report, but your program should smoothly handle given test-cases!!!***

**Lab 9 - Dungeons and Dragons, MCS 253P Style (see point-distributions below)**

A particular version of the Dungeons and Dragons game, the MCS 253P version, uses different sets of dice during different events. For example, during a player attack, perhaps two 4-sided dice and one 10-sided die are used. (The possibilities for types of die are 4-sided, 6-sided, 8-sided, 10-sided, and 12-sided, and assume any number of each can be used (within reason)).

When an event occurs, first the types and number of dice are determined by some algorithm. For the rest of the event, these are the only dice that will be considered.

Next, the following steps ensue:

1. the Dungeon Master (the person narrating the game and enforcing rules and fairness) determines ranges that will result in a *failure*, *draw*, or *success* for the player.
2. the Dungeon Master also determines the consequence of a *failure* and reward for *success* as integer values (*failures* as negatives, *successes* as positive, and *draws* defaulting to 0).
3. the player chooses a ***subset*** of the dice to be thrown (up to player). This will act towards the final roll.
4. finally, the player has one of two options:
   1. *withdraw* from the event and incur a penalty (increases with more dice preliminarily thrown)
   2. throw all the rest of the dice

**Lab 9.1 - Success Calculator (75pts)**

Design a program that utilizes dynamic programming to help the player make decisions by providing the the following information:

* probability of success given a currently rolled sum
* expected reward given the currently rolled sum

Input:

* currently rolled sum
* remaining dice to be rolled
* cutoffs for a *draw* (below the cutoff is implied as *failure*, and above the cutoff is implied as *success*)
* consequence for *failure* and reward for *success*

Output:

* probability of success given a currently rolled sum
* expected reward given the currently rolled sum

Example:

input format:

[currently rolled sum]

[number of 4-sided dice to be rolled][“ 6-sided “] ... [“ 12-sided “]

[inclusive lower bound for draw] [inclusive upper bound for draw]

[consequence for failure] [reward for success]



input:

2

2 0 1 0 0

7 9

-8 2

output:

p(success) = 74.2%

E = 1.23

**Lab 9.2 - Move Recommendation (25pts)**

Write a short description of how you would modify your program to ***output a suggested action*** for the user.

***>>> Be specific and feel free to be creative!!! <<<***

In your description, ***include any additional inputs*** you would obtain (***and why***), any ***additional outputs*** your would display, and ***other techniques/algorithms your program would use*** (feel free to use topics not covered in 253P)!