Course: COMPSCI 260
Name: Divya Koyyalagunta
NetID: dk160
Problem: 2
Problem Set: 5
Due: Fri 7 Nov 2016, 5pm
Using free extension (yes/no): no

Statement of collaboration and resources used (put None if you worked entirely without collaboration or resources; otherwise cite carefully): None
My solutions and comments for this problem are below.

PROBLEM 2

For both part (a) and part (b), I used a cumulative distributive function (cdf) to format event probabilities from an input such as $\{1: 0.3, 2: 0.4: 3: 0.3\}$, to $\{1: 0.3, 2: 0.7: 3: 1.0\}$. This allowed me to utilize the pseudorandom number generator in python, random.random, to find a number between 0.0 and 1.0. It is important to note that 1.0 will never be chosen from this method, so there are some edge effects to consider, since the last event in the input list will have a slightly less chance compare to the other events of being chosen regardless of the event probabilities. By using the random number generator, we can find the range of event probabilities that the random number fits into in the cumulative distributive function. So for example, if the random number chosen was 0.85, the event returned by the sampling method would be 2, since we find the maximum possible probability without going over the cdf value.

Part (a) uses the cdf and works in $O(n)$ time, because making the cdf takes $O(n)$, and searching through a list of length n takes average case and worst case n comparisons. In order to optimize, we can use binary search, start from the midpoint of the list, and search only the left and right halves based on whether the value we are trying to find (generated from random.random) is greater or less than the midpoint value we're at. We keep going through and updating the midpoint index, reducing the problem by half with each iteration. This makes the sampling process run in $O(\log_2 n)$ time, or just $O(logn)$ time.

The initialization step is the creation of the cumulative distributive function, which is created by iterating through each of the n event probabilities given as input, making the asymptotic run time of this step $O(n)$.