Pledge: I (we) have not received unauthorized aid on this assignment. I (we) understand the answers that I (we) have submitted. The answers submitted have not been directly copied from another source, but instead are written in my (our) own words.

1. [20 points]
Write a dynamic programming algorithm to solve the following variation on the Knapsack problem: You are given a collection of \( n \) items, where each item has both a size and a value. Each item has an infinite number of duplicates available. There is a size to the knapsack, \( K \). The goal is to select items with the greatest value possible, such that the sum of their sizes is less than or equal to \( K \).

2. [20 points]
Consider (yet again!!) the problem of searching for the position of \( X \) in an unsorted array, when we know that \( X \) appears exactly once in the array. Give a state-space lower bound argument that the worst-case lower bound for this problem is \( n - 1 \) comparisons. Hint: You may use an adversary as necessary to support the argument (typically this is done to rule out certain state transitions).

3. [20 points]
My MP3 player contains 1000 songs, and the average length for a song is 3 minutes (for a total of 3000 minutes or 50 hours of playtime). I like to use the “random” play feature to play songs at random. Given that I always play songs a random, what is the expected length of time that it takes before every song has been played at least once? You should come up with and solve a summation or recurrence relation to model this problem. (Note that the variance on the actual time required will be fairly high, but you are just determining the expected time.)

Warning: You will need to be fairly careful about the constants for this problem. I expect your answer to be approximate, but correct within 10%. You might come up with a summation for which I gave you a rough estimate of its closed-form cost. You might need to work it out more exactly than what I gave in the notes.