Exercises for chapter: Dynamic programming

1. For this example, draw up the recurrence relation between the expected scores. Prove that the relation is monotonically increasing (for decreasing index), and bounded above by 1, in fact with limit 1. Bonus: solve this relation explicitly.
2. A 'sparse matrix' is a matrix where a number of matrix elements are zero, sufficiently many that you do not want to store them. To compute the matrix vector product $y = Ax$ you then do not compute the full sum $y_i = \sum_j a_{ij}x_j$, but only those terms for which $a_{ij} \neq 0$. This sort of operation is easy enough to code, but is pretty inefficient in execution.
   Suppose that for small k the product with k consecutive matrix elements (that is $a_{ij}x_j + a_{ij+}x_{j+1} + \cdots + a_{ij+k-1}x_{j+k-1}$ can be executed more efficiently than doing it as k separate operations. For instance, suppose that with $k = 3$ the time per $a_{i.}x.$ reduced to .4 of the normal multiply time, that is, doing three consecutive multiplications as a block takes time 1.2, while doing them separately takes time 3.
   Now, if $a_{11} \neq 0$, $a_{12} = 0$, $a_{13} \neq 0$, the multiplication $a_{11}x_1 + a_{13}x_3$ takes time 2, but if we store the intermediate zero at a_{12}, the size 3 block multiplication takes time 1.2. This means that doing some superfluous operations (multiplying by zero) we can actually speed up the matrix-vector product.
   Let a pattern of nonzeros and reduction factors be given. The pattern stands for the locations of the nonzeros in a matrix row, for instance
   ```
   row = [1,0,0,1,0,1,1,0,0,1,0,0,0,1,1,0,1,1]
   redux = [1, .75, .4, .28]
   ```
   Formulate a principle of optimality for this problem, and write a dynamic programming solution for finding the covering of the sparse row that will lead to the shortest multiplication time. Tinker with the `redux` times (but make sure the n-th is more than $1/n$ in base-1 indexing) and see how the solution changes.