Give informal (at least) proofs for all answers. Grading will be on completeness and logical correctness, and if applicable, efficiency, as well as correctness. Out of 80 points.

**Backtracking and Dynamic Programming**

For the problems below, a correct backtracking recursive algorithm is given, with an example that illustrates how the backtracking algorithm works.

You need to: A: Give a time analysis for the backtracking algorithm, as an upper bound in $O$ notation (20 points), B: Convert the BT algorithm to a fast iterative dynamic programming algorithm (DP) (30 points), C: give a time analysis for your DP algorithm (20 points), and D: illustrate the array or matrix produced by your DP algorithm on the example given (10 points).

**Grade maximization**

We are taking a class with $k$ projects. We have $H$ hours to divide among the projects, and will spend an integer amount of time on each project. For every project $i$, we are given an array $G_i[0..H]$ so that, if we spend $h$ hours on project $i$, our grade for that project will be $G_i[h]$. Naturally, $G_i$ is increasing with $h$, and spending no time on an assignment gets 0 points, $0 = G_i[0] \le G_i[1] \le G_i[2] \le ... G_i[H]$.

We need to allocate $H$ hours among projects $1...n$, i.e., find non-negative integers $h_1, .. h_k$ with $\sum h_i = H$ in order to maximize $\sum_i G_i[h_i]$.

The following recursive algorithm for the general grade maximization problem is based on the possible answers to the question: How many hours do I devote to project 1? The possible answers are 0..$H$. The recursion just returns the best achievable total grade, not the assignment that achieves it.

$BTGM(H, G_1[0..H], .. G_k[0..H])$:

1. IF $n = 1$ return $G_1(H)$.
2. $BestGrade \leftarrow 0$.
3. FOR $h = 0$ to $H$ do:
   4. $ThisGrade \leftarrow G_1[h] + BTGM(H-h, G_2, .. G_k)$
   5. IF $ThisGrade > BestGrade$ THEN $BestGrade \leftarrow ThisGrade$.
6. Return $BestGrade$.

The tree of recursive calls for the example $k = 3, H = 3, G_1[0..3] = [0, 70, 80, 90], G_2[0..3] = [0, 30, 100, 100], G_3[0..3] = [0, 35, 75, 100]$ is on a separate page.