Problem 4. *(20 points)* Suppose you are given two strings x and y, of lengths m and n, respectively, a character match / mistmatch cost function $m(a, b)$, and a gap cost cost_g. Here, an optimal alignment is an alignment of *minimum cost* between x and y.

i. Give a dynamic program that computes the *number* of *optimal* global alignments between x and y for arbitrary $m(\cdot, \cdot)$ and cost_g. Your algorithm should run in $O(mn)$ time and space. Explain (no formal proof is necessary) why your algorithm is correct.

ii. Give a dynamic program that will compute the (distinct) **cost and path** of a *lowest cost* ***sub-optimal*** solution. That is, it should return a cost c' *strictly greater than* then optimal cost c^*, but there should not exist any c'' with $c^* < c'' < c'$. This represents the best solution that is strictly worse than optimal (i.e. the second-best solution). If there are no sub-optimal solutions (i.e. if every solution is optimal), then your algorithm should report this. Your algorithm should run in $O(mn)$ time and space.