1. [20 marks] **Formatting concrete poetry.** The problem of formatting concrete poetry is defined as follows:

Given a list of words $w_1, \ldots, w_n$ (where a *word* is a string of non-whitespace characters) and a list of line lengths $\ell_1, \ldots, \ell_m$, add line breaks and extra blank spaces to arrange the words into lines of exactly the required lengths in a way that minimizes $G$, the maximum number of blank spaces between any two consecutive words in any row.

When no additions of line breaks and extra blank spaces yields lines of the required lengths, the algorithm should output *No valid solution*.

(See *Programming Assignment 1* for a more detailed description of the problem and some examples.)

(a) [2 marks] One possible greedy algorithm for this problem involves putting as many words as possible in each line before starting to fill the following lines. Give an example showing that this greedy algorithm sometimes outputs *No valid solution* when some valid solutions do exist.

(b) [2 marks] Give a different counter-example to show that the greedy algorithm proposed in part (a) sometimes returns a valid solution that does not correctly minimize $G$.

(c) [16 marks] Design a dynamic programming algorithm that solves the problem of formatting concrete poetry described above. You should aim for an algorithm with time complexity $O(n^2 m)$. Clearly indicate what your subproblems are, and the order in which you solve them. Justify the correctness of your algorithm, and analyze its running time.