1. Describe a data structure that allows generating all secondary occurrences, given a list of primary occurrences, in $\mathcal{O}(\mathrm{occ} \log n)$ time and uses $\mathcal{O}(z \log n)$ space (or better).

2. Explain how to find the shortest superstring in $\mathcal{O}(2^n n^2)$ time using dynamic programming. For extra credit: keep the space polynomial in n at the expense of increasing the time by a factor of M, where M is the sum of lengths of all strings (hint: use inclusion-exclusion).

3. Consider a balanced SLP for $S[1..n]$ with r rules and integers b and g. We would like to be able to extract any $S[b - \ell..b + \ell]$ in $\mathcal{O}(\ell + \log g)$ time, for any $\ell \leq g$. Show how to store $2 \log r + \mathcal{O}(\log g)$ bits so that this is possible (hint: we need to store pointers to two nonterminals and an integer).

(2 points) 4. Consider a compacted trie T constructed for a collection of z strings. We would like to augment T so that we can implement a blind search for any $P[1..m]$ (recall that blind search considers only the first character on every edge when searching for $P[1..m]$ in T). Show how to implement this in $\mathcal{O}(\log z)$ time with a structure of size $\mathcal{O}(z)$ (hint: use Karp-Rabin fingerprints and some kind of tree decomposition). For extra credit: improve the time to $\mathcal{O}(\log m)$.