Homework 1 (related to LN1 and LN2)

The exercises do *NOT* requite long writings. Try to be precise and to the point.

1. Let C be a set consisting of n companies, and A be a set consisting of m applicants. Consider the set $C \times A$ of all ordered pairs of the form (c, a), where $c \in C$ and $a \in A$.
   (a) How many ordered pairs are there?
   (b) Explain your answer. (Keep your answer short, you can write your explanation in at most 2-3 short sentences).

   Solution: There are $n \cdot m$ ordered pairs. Each company can be paired with m applicants. There are n companies. Hence, there are $n \cdot m$ ordered pairs.

2. Solve all exercises in LN1.

   Solution of Exercise 1: Say $C = \{c_1, \ldots, c_n\}$ and $A = \{a_1, \ldots, a_n\}$. Company c_1 has n candidates to make an offer to. Once c_1 gave an offer, company c_2 has $n - 1$ applicants to make an offer to. Once c_1 and c_2 made their offers, company c_3 has $n - 3$ applicant to make an offer to. So, the total number of perfect matchings equals to

$$n \cdot (n-1) \cdot (n-2) \cdot \ldots 2 \cdot 1 = n!$$

   Solution of Exercise 2: One possible preference list could be the following. All companies rank the applicants the same way, for instance, $a_1 < a_2 < a_3$; and also all applicants rank companies the same way, $c_1 < c_2 < c_3$. The number of perfect matching (from the previous example) is always $3! = 6$ no matter what the preference list is. In the setting described, there is only one stable matching: $(c_1, a_1), (c_2, a_2), (c_3, a_3)$.

   Solution of Exercise 3: Easy. Left for tutorial.

   Solution of Exercise 4: Let M be the output of the algorithm. We already know (from the lecture) that no company is free. We also know that M is a match. If there is free applicant left, then the number of applicants m is greater than the number of companies n. This contradicts with the assumption that $n = m$.

   Solution of Exercise 5: Let n be the number of companies and m be the number of applicants. Then following the representation of the input as described in the lecture, we get the input size $n + m + 2n \cdot m$.
3. Solve all exercises in LN2.

   Solution of Exercise 1: For each $i = 1, \ldots, n$ do the following:
   Check if t_i contains one of the specified words.
   If t_i contains such a word then output t_i; Otherwise not.

The size of the input is n (the number of texts). However, if there is no bound on lengths of texts then the size cane defined as the sum of the lengths of the texts $t_1, \ldots, t_n$.

Solution of Exercise 2: Let C be the output of Merge(A, B) algorithm. Initially C is the empty list. Then after every iteration when a new item, say x, is added to C, it is the case that x no item in C is larger than x and not item outside of C is smaller than x. This is because A and B are sorted and due to the algorithm description. Hence, C is ordered.

Solution of Exercise 3: With each iteration the max value of a, b is at least twice less than the max value of a and b before the iteration. This implies that there are $O(size(input))$ iterations of the while loop. Each iteration takes a constant time. Hence the algorithm runs in linear time on $size(input)$.

4. Consider the GS-algorithm. Let M be the output of the algorithm. We know (from the lecture) that M is a stable matching.

   Say that an applicant x is *unlucky according to M* if the matching M assigns company c to the applicant x so that c is the worst ranked company in the applicant's preference list. Is it possible that *all* applicant are unlucky according to M? If so, then give such an example with 3 applicants and 3 companies. If not, explain your answer in brief.

   Solution: Assume the preference list of companies is this:
   $c_1: \quad a_1 < a_2 < a_3,$
   $c_2: \quad a_2 < a_3 < a_1,$
   $c_3: \quad a_3 < a_1 < a_2.$

   Assume that the preference list of the applicant is this:
   $a_1: \quad c_3 < c_2 < c_1,$
   $a_2: \quad c_1 < c_3 < c_2,$
   $a_3: \quad c_2 < c_1 < c_3.$
   Then the algorithm produces the following stable matching: $\{(c_1, a_1), (c_2, a_2), (c_3, a_3)\}$.

5. As above, let M be the output of the GS-algorithm. Assume that company c ranks an applicant x first; also assume that the applicant x, too, ranks c first. Does this imply that the pair (c, x) belongs to M? Answer the question as *false* or *true*, and explain your answer in brief.

   Solution: Yes. when it is c's turn to make an offer for the first time, c offers job to x. Since x ranks c highest x agrees to make an internship pair with c. From that point on any other company's offer to x will not be accepted by x.