\* 1.Find in the text T all occurences of the substrings which Hamming distance form the pattern P is at most *k*. Apply the dynamic programming approach.

a) *T* = ccacbaabccaccbcabccc, *P* = abcba, *k* = 2,

b) *T* = 000111011000101010111110,  *P* = 110010, *k* = 3.

\* 2. Find in the text T all occurences of the substrings which Levenshtein distance form the pattern P is at most *k*. Apply the dynamic programming approach.

a) *T* = aacacacbaabbbcbbcacc, *P* = cbbba, *k* = 3.

b) *T* = 010011101000010101011100 , *P* = 11100, *k* = 1

\* 3. Construct a non-deterministic automaton which detects each word of the set M in a text over the alphabet A .

a) A = {a, b, c}, M = {a, b, ba, bc, aaa, bab, ccc, abbc, abcc } .

b) A = {0, 1}, M = {10, 11, 101, 111, 1011, 1101, 10001, 10011, 10111, 11101, 11111 }.

\* 4. Construct a deterministic automaton which detects each word of the set M in a text over the alphabet A. M and A are the same as in the previous problem.

\* 5. Use the method of bit parallelism to construct the tables for simulation of the text search automaton which detects in the text *t* all occurences of substrings with Hamming distance *k*  from the pattern *p*.

a) *t* = abcbcaaccbbaa, *p* = bbac, *k* = 2,

b) *t* = accbbaaabcba, *p* = acbb, *k* = 2.

6. We define the reduced Levenshtein distance of strings X and Y to be the minimum number of edit operations which transform X to Y (or Y to X). Only edit operations Insert and Delete are considered in this definition. Describe an algorithm which will apply the Dynamic programming approach to compute the reduced Levenshtein distance between X and Y.

7. Describe an algorithm which will detect in a text all substrings which reduced Levenshtein distance from the given pattern is minimum possible. See the definition of the reduced Levenshtein distance in the previous problem.

8. We say that the Insert-distance of two strings X and Y is exactly *k* if and only if *k* is the minimum number of operations Insert applied to only one of the strings and which will make X and Y identical. If the identity cannot be achieved we define the Insert-distance of X and Y to be positive infinity. Construct a NFA which will accept all strings which Insert-distance from a given pattern is at most *k*.

9. Describe an algorithm based on the Dynamic programming approach which will detect in a text all occurences of strings which Insert-distance from a given pattern *p* is exactly *k*. See the definition of the Insert-distance in the previous problem.

10. Describe the changes which must be applied in the previous problem to obtain a similar algorithm which uses Delete-distance instead of Insert-distance.