**Given:**

The pattern-matching problem is as follows: Given a string, S, of text, and a pattern, P, ﬁnd the ﬁrst occurrence of P in S. Approximate pattern matching allows k mismatches of three types: (1) A character can be in S that is not in P. (2) A character can be in P that is not in S. (3) P and S can differ in a position. As an example, if we are searching for the pattern “textbook” with at most three mismatches in the string “data structures txtborpk”, we ﬁnd a match (insert an e, change an r to an o,delete a p). Give an O(MN) algorithm to solve the approximate string matching problem, where

M=| P|and N =| S|.

**Input and Output:**

The input to this project will ask for a value for k, number of acceptable errors, a file with the text to search through, and the pattern. The pattern will be larger than 0 and smaller than the size of the string in the file plus k. The program will then run the algorithm described below and return the matching pattern.

**Description:**

I believe the way to do this is by doing a backtracking algorithm. However, I am not quite sure how to go about writing one. I will start by placing the first character and claiming it as a count toward the error unless it matches perfectly. I then will move to the next character in the pattern and perform the same check. I will do this until the errors number is greater than allowed. In which case I will move the second to last character in the pattern to a different spot in the string to check for a deletion, again run this until the errors exceed allowed. I will then place the last character in an open space to show that the character is missing. I will then run the algorithm again. Except move to the last not backtracked character. The problem is I am not sure how to keep track of which character needs to be backtracked. For example I will need to be able to match textbook to tbook if there are three allowed errors. In which case I will need to be able to back track and put three characters into place holders. Another example I will need to account for is textboozzzk for three errors in which case I will need to move the k 3 spaces ahead. So each character can move the number of errors remaining in any direction. With this I would need a dynamic container that can hold the position of the first character and the position of the last character that works, a string of the remaining pattern to be checked, as well as the amount of errors so far. Once I find the pattern in the string, I need to return a true and look in the container for the position of the pattern match. If the pattern cannot be found the algorithm should return a false.

**Possibilities:**

Pattern “textbook”

Character in pattern matches character in string.

Characters do not match.

Character is not in String, ex txtbook

Character is not in Pattern ex teztbook

Character is extra ex texztbook

**Other Idea:**

I had another Idea to work backwards, I would compare the last character then when there isn’t a match and the errors are exceeded I would move the last character to the next space and add the second to last character of the pattern to the comparison. I would repeat this algorithm until a match is found. I am pretty sure this accounts for the first and last characters not being in the string. The problem is I feel this way involves a lot more checks per character so it will be slower than the algorithm described above and perhaps more accurate.