Question 1 (25 points) Figure 1 shows the perceptron algorithm, as described in lecture 20. Now say we alter the parameter update step to be the following:

\[
\begin{align*}
  \text{If } & (z_i \neq y_i) \\
  \mathcal{A} = \{ z : z \in \text{GEN}(x_i), z \neq y_i, W \cdot \Phi(x_i, z) \geq W \cdot \Phi(x_i, y_i) \} \\
  n = |\mathcal{A}| \\
  W = W + \Phi(x_i, y_i) - \frac{1}{n} \sum_{z \in \mathcal{A}} \Phi(x_i, z)
\end{align*}
\]

Show that the modified algorithm makes at most \( \frac{R^2}{\delta} \) updates before convergence, where \( R \) and \( \delta \) are as defined in the lecture (i.e., show that the convergence theorem that we described in lecture also holds for this algorithm). Hint: the proof is quite similar to the proof of convergence given in lecture.

<table>
<thead>
<tr>
<th>Inputs:</th>
<th>Training set ((x_i, y_i)) for (i = 1 \ldots n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization:</td>
<td>(W = 0)</td>
</tr>
<tr>
<td>Define:</td>
<td>(F(x) = \arg\max_{y \in \text{GEN}(x)} \Phi(x, y) \cdot W)</td>
</tr>
</tbody>
</table>
| Algorithm: | For \(t = 1 \ldots T, i = 1 \ldots n\) \\
| & \(z_i = F(x_i)\) \\
| & If \((z_i \neq y_i)\) \(W = W + \Phi(x_i, y_i) - \Phi(x_i, z_i)\) |
| Output: | Parameters \(W\) |

Figure 1: The perceptron algorithm, as introduced in lecture 20.

Question 2 (25 points) In lecture 17 we defined transduction PCFGs. For example, a transduction PCFG would assign a probability to a structure such as the following (see the lecture notes for more details):
The above structure can be considered to represent an English string \( e \), an English parse tree \( E \), a French string \( f \), and a French parse tree \( F \), in this case:

```
S
  NP VP VP NP
  D N Vi Vi D N
  the man sleeps asleeps
```

Say \( P(e, E, f, F) \) is the probability assigned to an \( e, E, f, F \) tuple by the transduction PCFG. Give pseudo-code for an algorithm that takes an \( e, E \) pair as input, and returns

\[
\arg \max_{f,F} P(e, E, f, F)
\]

**Question 3 (90 points)**

In this question you will implement code for IBM translation model 1. The files `corpus.en` and `corpus.de` have English and German sentences respectively, where the \( i \)’th sentence in the English file is a translation of the \( i \)’th sentence in the German file.

Implement a version of IBM model 1, which takes `corpus.en` and `corpus.de` as input. Your implementation should have the following features:

- The parameters of the model are \( T(f|e) \), where \( f \) is a German word, and \( e \) is an English word or the special symbol NULL. You should only store parameters of the form \( T(f|e) \) for \((f,e)\) pairs which are seen somewhere in aligned sentences in the corpus.
- In the initialization step, you should set \( T(f|e) = \frac{1}{n(e)} \) where \( n(e) \) is the number of different German words seen in German sentences aligned to English sentences that contain the word \( e \).
- Your code should run 10 iterations of the EM algorithm to re-estimate the \( T(f|e) \) parameters.

**Note: your code should have the following functionality. It should be able to read in a file, line by line, where each line has an English word, for example**

```
dog
eats
man
...```

For each line it should return a list of German words, together with probabilities \( T(f|e) \). The list of German words should contain all words for which \( T(f|e) > 0 \).