Problem 1

Write a MATLAB function \texttt{DocSimilarity(D,E)} which computes the “similarity” of text documents \(D\) and \(E\) using the vector model of documents. Specifically, the arguments \(D\) and \(E\) are each cell arrays of strings, each string being a word of the document, normalized to lower case. The function returns a number between 0 and 1, 0 meaning that the two documents have no two significant words in common, meaning that they have the identical significant words with the same frequency.

A word is considered “significant” if it has at least three letters and is not in the list of stop words provided at \texttt{GetStopwords.m} in the course code library, linked from the course web site. A stop word is a very common word that should be ignored.

Your function should execute the following steps.

- Let \(\text{LargeOdd}\) be any reasonably large odd number that is not very close to a power of 256. 10,000,001 will do fine.
- Load in the cell array of stop words from \texttt{GetStopwords.m}
- Create three sparse vectors \(\vec{S}, \vec{D}, \vec{E}\) of size \(\text{LargeOdd}\), as follows: For every word \(W\), let \(i=\text{hash}(W,\text{LargeOdd})\). You can find a hash function \texttt{hash.m} in the course code library. Then
  - \(\vec{S}[i] = 1\) if \(W\) is on the list of stop words.
  - \(\vec{D}[i] = \) the number of occurrences of \(W\) in \(D\), if \(W\) is significant.
  - \(\vec{E}[i] = \) the number of occurrences of \(W\) in \(E\), if \(W\) is significant.

(Create \(\vec{S}\) first, then use it for a quick test for whether words in the documents are significant.)

\(\vec{D}\) and \(\vec{E}\) are the document vectors (we omit the inverse document frequency).

- Return the quantity \(\vec{D} \cdot \vec{E} / |\vec{D}| |\vec{E}|\)

For instance,

\begin{verbatim}
>> D = { 'how', 'much', 'wood', 'could', 'a', 'woodchuck', 'chuck', ...
       'if', 'a', 'woodchuck', 'could', 'chuck', 'wood' };
>> E = { 'all', 'the', 'wood', 'that', 'a', 'woodchuck', 'could', ...
       'if', 'a', 'woodchuck', 'could', 'chuck', 'wood' };
>> DocSimilarity(D,E)
ans =
    0.9245
\end{verbatim}
Note that the only significant words in these two texts are “chuck”, “much”, “wood”, and “wood-chuck”.

You don’t have to worry about hash collisions, because they are very infrequent, and the technique is completely imprecise in any case.

**Problem 2**

The points on an ellipse with center \( p = (p_x, p_y) \) with semi-axes of length \( a \) and \( b \), where the semi-axis of length \( a \) is oriented at angle \( \theta \) from the \( x \) axis, is given by the parameterized curve

\[
(p_x, p_y) + \langle a \cos(\theta) \cos(\phi) - b \sin(\theta) \sin(\phi), a \sin(\theta) \cos(\phi) + b \cos(\theta) \sin(\phi) \rangle, \quad \phi \in [0, 2\pi]
\]

**Part A**

Write a MATLAB function `Ellipse(p,a,b,theta)` where \( p \) is a column vector; \( a, b, \) and \( \text{theta} \) are floating points and \( \text{theta} \) is in radians. The function

- i. Computes the 101 points on the ellipse corresponding to the values \( \phi = (i - 1) \ast 2\pi/100, i = 1 \ldots 101 \) and puts these coordinates in a \( 2 \times 101 \) matrix, which it returns as its value;
- ii. Draws a picture of the ellipse in black.

For instance, the function call `Ellipse([2;1],3,1,pi/6)` generates the figure shown below, and returns a \( 2 \times 101 \) matrix whose first 7 columns are:

**Columns 1 through 7**

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5000</td>
<td>2.5514</td>
<td>2.5967</td>
<td>2.6357</td>
<td>2.6682</td>
<td>2.6942</td>
<td>2.7135</td>
</tr>
</tbody>
</table>
You should try to write the function as “straight-line” code, with no explicit loops at the level of the MATLAB code. Hint: Use the `repmat` function to turn the point $p$ into a $2 \times 101$ array. Of course the MATLAB built-in functions you are calling have such a loop in the background, but these are presumably much more efficiently implemented than an explicit loop in the MATLAB code. There will be a penalty of 2 points out of 100 (total for the entire assignment) for writing an explicit loop.

For all assignments involving drawing geometric pictures, you should use the plotting call `axis equal` to use equal scales on the x and y axes; otherwise, the picture tends to come out visually squooshed.

**Part B**

Write a MATLAB function `TwoEllipses(p1,a1,b1,theta1, p2,a2,b2,theta2)` that

i. Draws the two ellipses in black.

ii. Draws a blue line between the two points on the two ellipses that are furthest apart, and a red line between the two points that are closest together. (Break ties arbitrarily.) If the two points that are closest together are less than 0.1 apart, then it draws an asterisk there rather than a red line.

iii. Returns two values: the distance between the two closest points and the distance between the two most distant points.
When you want to do multiple calls to `plot` in the same figure, you need to call `hold on`; otherwise, each call to `plot` erases the previous ones.

You may use explicit loops in this part of the assignment.

For example the function call

```matlab
>> [Short,Long]=TwoEllipses([2;1],3,1,pi/6,[3;-1],2,2.5,3*pi/4)
```

```matlab
Short =
    0.0437
Long =
    6.3588
```

generates the following picture: