For lab 2 you implemented a method for finding word similarities from their embeddings using a binary search tree to enable efficient search. For this lab, your task is to implement the same methods using a B-tree and experiment with various values of $t$, trying to determine which value yields the best performance. Feel free to use the code provided in the class webpage as a starting point.

Compare the running times of your program using B-Trees with multiple choices of $t$ and also compare with the running time of your binary search tree. For each value of $t$, display also the resulting height of the tree.

In order to obtain meaningful values, you need to find similarities of a large number of word pairs. To do this, provide a list of words and compute the similarity between each pair of words in the list (thus for a list with $n$ words, you will compute $n^2$ similarities). Store the similarities in a 2D array $S$, where $S[i][j]$ is the similarity between word $i$ and word $j$. Notice that $S[i][j] == S[j][i]$ and that $S[i][i] == 1$ (do you see why?). Then, for each word $w$ in the list, display the word that is most similar to $w$ (except $w$ itself). To compute running times, do not consider the times required to perform input and output, only consider the times required to store the data in the tree and to compute the similarity matrix.

As usual, write a report describing your work. Present tables illustrating actual running times and discuss the effect of your choice of $t$ in the resulting running times.