1. **Simulating improved breakpoint reversal sort.**
   Perform the improved breakpoint reversal sort algorithm (page 28 at lecture slides) with $\pi = 3 \ 4 \ 6 \ 5 \ 8 \ 1 \ 7 \ 2$ and show all intermediate permutations. Is this the optimal solution to this instance of reversal sorting problem?

2. **Transforming circular genome.**
   Devise an approximation algorithm to sort a circular genome by reversals (i.e., transform it to the identity circular permutation). Evaluate the algorithm’s performance guarantee.

3. **Implementing improved breakpoint reversal sort.**
   Write a Python program that implements improved breakpoint reversal sort and analyse the running time of your implementation.

4. **Shortest approximate superstring.**
   Let $S = S_1, S_2, \ldots, S_n \subseteq \Sigma^*$ be a set of strings from alphabet $\Sigma$. Given a threshold parameter $k$, an approximate superstring of $S$ is defined as a string $T$ such that for each $S_i \in S$ it holds $d_H(S_i, T[j_i \cdots j_i + |S_i| - 1]) \leq k$ for some $j_i$, where $d_H(\cdot)$ denotes the Hamming distance.
   
   A greedy approximation algorithm for finding the shortest approximate superstring can be derived as follows. Let an approximate overlap of $A = \alpha \gamma, B = \gamma' \beta \in S$ be pair of strings $(\gamma, \gamma')$ such that $d_H(\gamma, \gamma') \leq k$ and the length of the overlap $|\gamma| = |\gamma'|$ is maximum among all ways to to write $A$ and $B$ in parts $A = \alpha \gamma$ and $B = \gamma' \beta$. Iterate the following until there is only one string in set $S$: (1) Choose $A = \alpha \gamma, B = \gamma' \beta \in S$ with maximum approximate overlap; (2) remove $A$ and $B$ from $S$ and insert $\alpha \gamma \beta$ into $S$.
   
   Simulate the above greedy algorithm with $k = 1$ on the set \{ACACGATC, ATGACAAA, TAATAAGA, CAGGATCA\}.
   
   Is the solution of your simulation a valid approximate superstring? Does the algorithm always find a valid approximate superstring? If not, give a modification so that it does.