1. **Assignment 4:** Shadi Zidany
2. **Search Trees Compared:**
   - AVL Tree
   - Binary Search Tree (BST)
   - Splay Tree
3. **Key Sets Description:**
   Two key sets are used in the experiment:
   - **Set2: n2 = 200,000:** Constructed by taking a vector of integers [0, 1, 2, …, (n2 – 1)] and randomly shuffling it.
   - **Set1: n1 = half the size of n2:** Derived from the shuffled vector v such that for each index i from 0 to (n1 – 1) the key at the i-th position is computed as v[v[i]].

   The keys from each set are inserted in the order they are generated.
4. **Research Question:**
   The purpose of the experiment is to investigate how the insertion and in-order traversal performance of different search trees (self-balancing vs. non-balancing) compare when subjected to increasing dataset sizes. This comparison helps in understanding the trade-offs between faster insertions and the overhead of maintaining tree balance.
5. **Program Timing Results:** in milliseconds

Operation	AVL Tree	BST	Splay Tree
Insert set1	52.67	20.37	29.12
Insert set2	97.37	44.45	65.06
Traversal set1	1.42	1.74	2.01
Traversal set2	3.35	4.02	4.41

6. **Observations:**
   - The BST shows the fastest insertion times for both sets, while the AVL and Splay trees take longer, likely due to the additional work required for balancing.
   - In-order traversal times are low across all trees, with minor variations, indicating that once built, the traversal performance is less sensitive to tree type.
7. **Inferences:**
   The data suggest that while self-balancing trees (AVL and Splay) incur extra insertion overhead to maintain balanced structures, a standard BST—assuming the input data is random—can achieve faster insertions. However, the benefits of balanced trees may become more pronounced in scenarios involving worst-case inputs or when additional operations (like deletion or search) are considered.
8. **Additional Notes:**
   This experiment underscores the importance of aligning data structure choice with application needs. Challenges included ensuring that the random data sets adequately tested balancing overhead. Future experiments might incorporate a broader range of key distributions (e.g., nearly sorted data) and additional operations (such as deletions) for a more comprehensive performance evaluation.