1. We have discussed the Gale-Shapely algorithm for stable matching in class at a very high level. This algorithm takes \( O(n^2) \) time if implemented properly—\( O(1) \) time for each of the \( O(n^2) \) iterations. The running time will be much worse if the algorithm is not implemented properly.

   (a) (20 pts) Discuss how to implement the algorithm using elementary data structures such as arrays, linked lists, stacks, and queues to represent preference lists, the list of unmarried men, etc. The resulting algorithm should take \( O(n^2) \) time.

   (b) (15 pts) Prove that the running time of the algorithm is indeed \( O(n^2) \). You may use the fact that there are \( O(n^2) \) iterations as proved in class, but do not forget to consider the time required to initialize data structures.

2. (35 pts) Given a set \( S \) of \( n \) integers, we are interested in supporting the following operations:

   \( \text{Find}(S, x) \) : Decide whether \( x \in S \).

   \( \text{Range-Find}(S, a, b) \) : Report all elements \( x \in S \) with \( a \leq x \leq b \).

   \( \text{Predecessor}(S, x) \) : Given an element \( x \in S \), report the next smaller element in \( S \).

   \( \text{Insert}(S, x) \) : Add the element \( x \) to \( S \).

   \( \text{Delete}(S, x) \) : Delete the element \( x \) from \( S \).

Make a table listing the costs of these operations on each of the following data structures that can be used to represent \( S \):

- An unsorted array of elements,
- a sorted array of elements,
- a linked list,
- a doubly-linked list,
- a hash table, and
- a red-black tree.

You do not need to argue why these operations have the costs you state, but I assume that you would be able to if asked. This knowledge is an essential foundation we will build on throughout the course. You should have seen all of these data structures before (except, perhaps the red-black tree); the purpose of this question is to review what you know about these data structures. If you are not familiar with red-black trees then you may find the information on them through any
means other than asking someone to answer the question for you (i.e. the internet, library, etc.)
. This is the one exception to the usual rule of not using outside sources.

For linked and doubly-linked lists, choose whether or not to keep it sorted. Make the choice that
gives you the best overall performance of the data structure. (If your choice makes one or more
operations slower than if you had chosen the other option, but none of the other operations gets
faster, you have made the wrong choice.)

For hash tables, state the expected and the worst-case cost of each operation and indicate clearly
which is which.