Problem 1. [30 points]
(a) In the following table, show the worst-case running times for the best algorithms for implementing the given operations for the dynamic data structure representations listed in the leftmost column. In each case, an argument of (S, key) implies we are passing a key to the procedure and (S, obj) implies we are passing a pointer to the object. The procedure Min(S) takes as input a set S, and returns the minimum element in the set. The procedure Sort(S) takes as input a set S, and returns a new linked list containing the elements of S in sorted order.

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Search(S,key)</th>
<th>Insert(S,key)</th>
<th>Delete(S,obj)</th>
<th>Min(S)</th>
<th>Sort(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsorted Array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sorted Array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unsorted Linked List</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sorted Linked List</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unsorted Doubly Linked List</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sorted Doubly Linked List</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min Heap</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) For each of the data structures above, explain in a few sentences how you’d implement the Min and Sort procedures.
Problem 2.  [20 points]
In order to asymptotically analyze the running time of an algorithm, one has to think about how the data will be represented and manipulated in an implementation of the algorithm, so as to bound the number of computational steps it takes. In this problem, you will use lists and arrays to implement the different high-level operations in the Gale-Shapley algorithm presented in Chapter 1, and shown in Figure 1.

---

Initially all $m \in M$ and $w \in W$ are free
While there is a man $m$ who is free and hasn’t proposed to every woman
Choose such a man $m$
Let $w$ be the highest-ranked woman in $m$’s preference list to whom $m$ has not yet proposed
If $w$ is free then
$(m, w)$ become engaged
Else $w$ is currently engaged to $m’$
If $w$ prefers $m’$ to $m$ then
$m$ remains free
Else $w$ prefers $m$ to $m’$
$(m, w)$ become engaged
$m’$ becomes free
Endif
Endif
Endwhile
Return the set $S$ of engaged pairs

---

Figure 1: The Gale-Shapley algorithm.

(a) For each of the following high-level operations, explain which data structure would you use to optimize the operation, and its associated worst case running time.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Data Structure</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representing the $n$ men and women</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Representing the preference list of the men</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Representing the preference list of the women</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Representing the set $S$ of engagements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finding the next free man ($m$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For a man ($m$), finding the next woman ($w$) to propose to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For $w$, checking if she is engaged to $m’$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For $w$, deciding whether she prefers $m$ over $m’$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Rewrite the algorithm, using lists and arrays only.
(c) Analyze the worst case running time for your algorithm.
Problem 3. [35 points] KT Chapter 1 Problem 5.
Consider a version of the stable matching problem where there are \( n \) men and \( n \) women as before. Assume each man ranks the women (and vice versa), but now we allow ties in the ranking. In other words, we could have a man that is indifferent two women \( w_1 \) and \( w_2 \), but prefers either of them over some other woman \( w_3 \) (and vice versa). We say a woman \( w \) prefers man \( m_1 \) to \( m_2 \) if \( m_1 \) is ranked higher on the \( w \)'s preference list and \( m_1 \) and \( m_2 \) are not tied.

With indifferences in the rankings, there could be two natural notions for stability. And for each, we can ask about the existence of stable matchings, as follows.

(a) **Strong Instability.** A strong instability in a perfect matching, \( S \), consists of a man \( m \) and a woman \( w \), such that each of \( m \) and \( w \) prefers the other to their partner in \( S \). **Does there always exist a perfect matching with no strong instability?** Either give an example of a set of men and women with preference lists for which every perfect matching has a strong instability; or give an algorithm that is guaranteed to find a perfect matching with no strong instability.

(b) **Weak Instability.** A weak instability in a perfect matching, \( S \), consists of a man \( m \) and a woman \( w \), such that their partners in \( S \) are \( w' \) and \( m' \), respectively, and one of the following holds:
* \( m \) prefers \( w \) to \( w' \), and \( w \) either prefers \( m \) to \( m' \) or is indifferent between these two choices; or
* \( w \) prefers \( m \) to \( m' \), and \( m \) either prefers \( w \) to \( w' \) or is indifferent between these two choices.

In other words, the pairing between \( m \) and \( w \) is either preferred by both, or preferred by one while the other is indifferent. **Does there always exist a perfect matching with no weak instability?** Either give an example of a set of men and women with preference lists for which every perfect matching has a weak instability; or give an algorithm that is guaranteed to find a perfect matching with no weak instability.

Problem 4. [15 points]
From your understanding of min heaps:

(a) What are the minimum and maximum numbers of elements in a heap of height \( h \)?

(b) Show that an \( n \)-element heap has height \( \lfloor \log n \rfloor \).

(c) Show that, with the array representation for storing an \( n \)-element heap, the leaves are the nodes indexed by \( \lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, ..., n \).

Problem 5. [Bonus 15 points] Challenge Problem. Solve this problem correctly, and you will get 15 bonus points towards this assignment or the next assignment.

KT Chapter 1 Problem 8. The basic question is: Can a man or a woman end up better off by lying about his or her preferences? More concretely, we suppose each participant has a true preference order. Now consider a woman \( w \). Suppose \( w \) prefers man \( m \) to \( m' \), but both \( m \) and \( m' \) are low on her list of preferences. Can it be the case that by switching the order of \( m \) and \( m' \) on her list of preferences (i.e., by falsely claiming that she prefers \( m' \) to \( m \)) and running the algorithm with this false preference list, \( w \) will end up with a man \( m'' \) that she truly prefers to both \( m \) and \( m' \)? (We can ask the same question for men, but will focus on the case of women for purposes of this question.)

Resolve this question by doing one of the following two things:

(a) Give a proof that, for any set of preference lists, switching the order of a pair on the list cannot improve a woman’s partner in the Gale-Shapley algorithm; or

(b) Give an example of a set of preference lists for which there is a switch that would improve the partner of a woman who switched preferences.