The same general rules as for Assignment 1 apply: see the course website.

1 Sorting (3 points)

(i) Which of the following sorting algorithms are stable: insertion sort, merge sort, heapsort, and quicksort? State any assumptions that you make.

(ii) Give a simple scheme which makes any sorting algorithm stable. How much additional time is needed if your scheme is used?

2 Choosing a good data structure (3 points)

(i) Complete the following table:

<table>
<thead>
<tr>
<th>Data structure</th>
<th>Insert</th>
<th>Search</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linked List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash Table</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Search Tree</td>
<td></td>
<td></td>
<td>$\mathcal{O}(\log n)$</td>
</tr>
</tbody>
</table>

Make sure you fill in the average case runtime, and not the worst case. State any assumptions you make, if anything is unclear.

(ii) With the help of your newly created table, describe a scenario for each data structure, where that particular data structure would be the best choice of the three. Justify your answer, and relate it to the table.

\(^1\)Assume uniform hashing and collision resolution by chaining.
3 Array concatenation (4 points)

The concatenation $A_1 \land A_2$ of two arrays $A_1 = \langle e_1, e_2, ..., e_n \rangle$ and $A_2 = \langle f_1, f_2, ..., f_m \rangle$ is defined as: $A_1 \land A_2 = \langle e_1, e_2, ..., e_n, f_1, f_2, ..., f_m \rangle$. $A \land B$ can be implemented by the following $\Theta(length[A] + length[B])$ time procedure:

\textsc{Array-Concat}(A, B)
1. Create a new array $C$, with $length[A] + length[B]$ slots
2. \textbf{for} $i \leftarrow 1$ \textbf{to} $length[A]$
3. \hspace{1em} \textbf{do} $C[i] \leftarrow A[i]$
4. \textbf{for} $i \leftarrow 1$ \textbf{to} $length[B]$
5. \hspace{1em} \textbf{do} $C[i + length[A]] \leftarrow B[i]$
6. \textbf{return} $C$

Now, suppose we want to concatenate $n$ arrays to each other, ie we wish to compute $A_1 \land A_2 \land ... \land A_n$. For convenience, we assume that all arrays have length $m$.

(i) What is the asymptotic runtime of this computation if $\land$ is implemented by \textsc{Array-Concat}?

(ii) Give a more efficient method for computing $A_1 \land A_2 \land ... \land A_n$. 
