Instructions:

1. This is a closed book exam. Do not use any notes, books, or neighbors except your one page, two side, cheat sheet which MUST be turned in with your exam.

2. Show your work. Partial credit will be given. Grading will be based on correctness and clarity.

3. You have 60 minutes to complete the exam. Watch your time appropriately. You should take about 15 minutes per question section.

Integrity: The Aggie Honor Code is An Aggie does not lie, cheat, or steal or tolerate those who do. Upon accepting admission to Texas A&M University, a student immediately assumes a commitment to uphold the Honor Code, to accept responsibility for learning, and to follow the philosophy and rules of the Honor System. Students will be required to state their commitment on examinations, research papers, and other academic work. Ignorance of the rules does not exclude any member of the TAMU community from the requirements or the processes of the Honor System.

I agree to uphold this commitment and produce original work in this exam.

Signature: ________________________________

DO NOT BEGIN THE EXAM UNTIL INSTRUCTED TO DO SO. GOOD LUCK!

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>received</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>
1. (20 points, 2 points each) Answer the following questions.

(a) True or False: A map/dictionary implemented as a linked list (e.g., log file) has very good space usage, but might not have very fast operation times.

(b) True or False: A map/dictionary implemented as a direct-address table has very fast operation times, but might not have very good space usage.

(c) Assume there are \( N \) slots in your hash table, and that there are \( n \) data items stored in your hash table. In hashing with chaining, the space usage will be \( \Theta(n+N) \). In open-addressing hashing, the space usage will be \( \Theta(N) \).

(d) Consider a binary search tree \( T \) storing \( n \) (key,element) pairs. The time for a \texttt{find}(k) operation is \( O(\log n) \) in the best case (use asymptotic notation) and the time for a \texttt{put}(k,v) operation is \( O(n) \) in the worst case (use asymptotic notation).

(e) Consider an AVL tree \( T \) storing \( n \) (key,element) pairs, and let \( h \) denote the height of \( T \). In the best case, \( h \) is \( O(\log n) \) (use asymptotic notation) and in the worst \( h \) is \( O(\log n) \) (use asymptotic notation).

(f) Suppose a recursive divide-and-conquer algorithm partitions the original problem into 5 subproblems, each of size \( \frac{n}{3} \), and that it spends \( \Theta(n) \) time to partition the original problem into subproblems and \( \Theta(n \log n) \) time to combine the subproblem solutions into the solution to the original problem, where \( n \) is the input size.

Write down the recurrence relation showing the running time of this algorithm:
\[
T(n) = 5T\left(\frac{n}{3}\right) + \Theta(n \log n).
\]

(g) Merge sort is an algorithm that applies the divide-and-conquer paradigm. Merge sort partitions the original problem of size \( n \) into \( 2 \) subproblems and spends \( O(n) \) time total to partition the problem into subproblems and to combine the subproblem solutions into a solution to the original problem.

(h) Quicksort is an algorithm that applies the divide-and-conquer paradigm. Quicksort partitions the original problem of size \( n \) into \( 2 \) subproblems and spends \( O(n) \) time total to partition the problem into subproblems and to combine the subproblem solutions into a solution to the original problem.

(i) True or False: Given a sorted sequence of \( n \) elements stored in an array, the selection problem can be solved in \( O(1) \) time.

(j) Radix sort internally applies Bucket sort to lexicographically order a set of tuples. Assuming each of the \( n \) tuples have \( d \) dimensions and there are \( N \) buckets, the running time of radix sort is \( O(d(n+N)) \).
2. (20 points) **Hashing.** Consider inserting the following keys, in this order, into a hash table of size \( m = 11 \).

**keys to insert (in this order): 2, 3, 13, 14, 24, 25**

(a) (4 points) Suppose you use chaining with the hash function \( h(k) = k \mod 11 \). Illustrate the result of inserting the keys above using chaining.

(b) (4 points) What is the (i) expected time and the (ii) worst-case running time for a find operation on a hash table of size \( N \) that contains \( n \) items, where collisions are resolved by chaining? Clearly state any assumptions.

(i) \( O\left(\frac{n}{N}\right) \)

(ii) \( O(n) \)

(c) (4 points) Suppose you use open addressing with the primary hash function \( h(k) = k \mod 11 \). Illustrate the result of inserting the keys above using linear probing, i.e., \( h(k, i) = (h(k) + i) \mod 11 \).

(d) (4 points) Suppose you use open addressing with hash functions \( h_1(k) = k \mod 11 \) and \( h_2(k) = 1 + (k \mod 10) \). Illustrate the result of inserting the keys above using double hashing, i.e., \( h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod 11 \).

(e) (4 points) What is the (i) expected time and the (ii) worst-case running time for inserting \( n \) items into an initially empty hash table of size \( N \) when using open addressing? Clearly state any assumptions.

(i) \( O\left(nN/(N-n)\right) \)

(ii) \( O(n^2) \)
3. (30 points) **Binary Search Trees (BST) and AVL Trees.**

(a) (5 points) Draw the BST that would result from inserting the keys (5, 16, 22, 45, 2, 10) (in this order) into an initially empty BST. Show explicitly any intermediate restructuring that is required.

```
  5
 /   \
2     16
     /   \
    10   22
         /   \n        45   
```

(b) (5 points) Draw the resulting BST when the item with key 3 is removed from the BST shown below.

```
  1
 /  \
3   2
   /  \
  6   5
 /  \
4   8
```

```
  1
 /  \
4   2
   /  \
  6   5
   /  \
  8   
```

(c) (5 points) Draw the AVL tree that would result from inserting the keys (5, 16, 22, 45, 2, 10) (in this order) into an initially empty AVL tree. Show explicitly any intermediate restructuring that is required.

```
  5
 /   \
2     16
     /   \
    22   
```

```
  16
 /   \
 5    22
```

```
  16
 /   \
 5    22
```

```
  16
 /   \
 5    22
```

(d) (5 points) Draw the resulting AVL when the item with key 2 is removed from the AVL shown below.

```
  3
 /  \
2   7
   /  \
  1   8
 /  \
4   6
```

```
  3
 /  \
5   7
 /  \
1   8
```

```
  3
 /  \
5   7
 /  \
1   8
```

```
  3
 /  \
5   7
 /  \
1   8
```

```
  3
 /  \
5   7
 /  \
1   8
```

5
(e) (5 points) Which ADTs are a search tree meant to implement most efficiently?

Ordered Map, Ordered Dictionary

(f) (5 points) Name two other self-balancing search trees besides AVL tree.

(i) Splay Tree

(ii) Red-black Tree
4. (30 points) Assume you are given a function $\text{MAGIC\_MERGE}(S_1, S_2, S_3)$ which takes in three sorted sequences of size $\frac{n}{3}$ and combines them into a single sorted sequence using a less-than comparison. Additionally assume, that you can algorithmically access the third of any sequence with a simple function.

(a) (10 pts) Design a comparison based sorting algorithm, called $\text{MAGIC\_SORT}(S)$, using the function $\text{MAGIC\_MERGE}(S_1, S_2, S_3)$.

**Algorithm** $\text{MAGIC\_SORT}(S)$

**Input:** Sequence $S$ of items.

```
if $S\text{.size}() > 1$ then
    $S_1 \leftarrow \text{MAGIC\_SORT}(1\text{st third of } S)$
    $S_2 \leftarrow \text{MAGIC\_SORT}(2\text{nd third of } S)$
    $S_3 \leftarrow \text{MAGIC\_SORT}(3\text{rd third of } S)$
    $S \leftarrow \text{MAGIC\_MERGE}(S_1, S_2, S_3)$
return $S$
```

(b) (5 points) State the generic recurrence relation for $\text{MAGIC\_SORT}$ on a sequence of size $n$. Assume the time for splitting the sequence is $O(1)$ and express the time for merging using a general function.

\[
T(n) = 3T\left(\frac{n}{3}\right) + M(n)
\]

(c) (5 points) Assume the running time of $\text{MAGIC\_MERGE}(S_1, S_2, S_3)$ is $\Theta(n)$. State and solve the recurrence relation for the asymptotic running time of $\text{MAGIC\_SORT}$.

\[
T(n) = 3T\left(\frac{n}{3}\right) + \Theta(n)
\]

\[
T(n) = 3(T\left(\frac{n}{9}\right) + \Theta\left(\frac{n}{3}\right)) + \Theta(n)
\]

\[
\vdots
\]

\[
T(n) = \sum_{i=0}^{\log_3 n} \frac{n}{3^i}
\]

\[
T(n) = O(n \log_3 n)
\]
(d) (5 points) Assume the running time of magic_merge($S_1, S_2, S_3$) is $\Theta(1)$. State and solve the recurrence relation for the asymptotic running time of magic_sort.

\[
T(n) = 3T\left(\frac{n}{3}\right) + \Theta(1)
\]
\[
T(n) = 3(T(\frac{n}{9}) + \Theta(1)) + \Theta(1)
\]
\[\vdots\]
\[
T(n) = \sum_{i=0}^{\log_3 n} 1
\]
\[
T(n) = O(\log_3 n)
\]

(e) (5 points) Is it possible for magic_merge($S_1, S_2, S_3$) to run in $O(1)$ time? Why or Why not?

No. The lower bound for comparison-based sorting is $\Omega(n \log n)$ which is asymptotically more than the derived equation in part (c). Additionally, lower bound for non-comparison-based sorting is $\Omega(n)$.

(f) (Bonus 10 points) Describe how you would modify magic_sort($S$) to run on a parallel computer and give (but do not solve) the recurrence relation for the time complexity assuming that magic_merge($S_1, S_2, S_3$) runs in $O(\log n)$ and you have as many processors as you can use, i.e., $p = n$. (Hint: Parallel computers do things at the same time. So if two parts of an algorithm happen at once, I only count the one which takes the longest as its time complexity.)

Do each recursion in parallel on $\frac{n}{3}$ processors per third.

\[
T(n) = T(\frac{n}{3}) + O(\log n)
\]