Name: _________ Key

Instructions:

1. There are test questions on the front and the back of each sheet.

2. This is a closed book exam. Do not use any notes, books, or neighbors except your one page, two side, handwritten, cheat sheet. You may also have an additional blank page as scratch paper.

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

4. You have 75 minutes to complete the exam. Watch your time appropriately. You should take about 15-20 minutes per question section.

Integrity: The University of Richmond’s Honor Code is “We, the students of the University of Richmond, shall promote and uphold a community of integrity and trust.” Upon accepting admission to University of Richmond, 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 Richmond community from the requirements or the processes of the Honor System.

I agree to uphold this commitment and produce original work in this exam, i.e., I will not cheat nor will I consciously let anyone cheat.

Signature: ________________________________

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

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>received</th>
<th>possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Bonus</td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. (20 points, 2 points each) Answer the following questions.

(a) True or False: Sorting with a Priority Queue (PQ-sort) consists of performing \( \frac{n}{2} \) \text{insert}(e) commands followed by \( \frac{n}{2} \) \text{removeMin}() operations.

(b) When a PQ is implemented with an unsorted sequence, the PQ-sort algorithm is referred to as _____ Selection Sort _____ (a type of sorting algorithm). When a PQ is implemented with a sorted sequence, the PQ-sort algorithm is referred to as _____ Insertion Sort _____ (a type of sorting algorithm). Both sorts run in time \( O(n^2) \).

(c) When a PQ is implemented with a binary heap, the \text{insert}(k, v) \) operation and the \text{removeMin}() \) operations both take \( O(\log n) \) time. In this case, the PQ-sort algorithm is referred to as _____ Heap Sort _____ (a type of sorting algorithm) and runs in time \( O(n \log n) \).

(d) The preferred implementation of a Priority Queue is a(n) _____ array-based heap _____ (be as specific as possible). Because of its layout in memory, it yields performance gains from the phenomena of _____ locality _____.

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

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

(g) 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 \( O(n+N) \). In open-addressing hashing, the space usage will be \( O(N) \).

(h) Consider a binary search tree \( T \) storing \( n \) (key,element) pairs. The time for a \text{get}(k) \) operation is _____ \( O(n) \) _____ in the worst case when \( k \) is unique to \( T \), and the time for a \text{put}(k,v) \) operation is _____ \( O(\log n) \) _____ in the best case when \( k \) is unique to \( T \).

(i) Consider an AVL tree \( T \) storing \( n \) (key, value) pairs. The height is _____ \( O(\log n) \) _____ in the worst case, and the \text{put}(k,v) \) and \text{remove}(k) \) operations require _____ \( O(\log n) \) _____ tri-node restructurings in the worst case.

(j) When implementing the Sorted Multi-Map ADT, a _____ red-black tree _____ is the preferred implementation, and when implementing the Set ADT, a _____ Hash map with chaining _____ is the preferred implementation.
2. (20 points) **Hashing.** Consider inserting the following keys, in this order, into a hash table of size $N = 9$.

**keys to insert (in this order):** 5, 6, 7, 15, 16, 17

(a) (6 points) Suppose you use chaining with the hash function $h(k) = 2k \mod N$. Illustrate the result of inserting the keys above using chaining.

```
H:  0  1  2  3  4  5  6  7  8
   5  6  7  15 16 17
```

(b) (4 points) What is the (i) expected time and the (ii) worst-case running time for a get operation on a hash table of size $N$ that contains $n$ items, where collisions are resolved by chaining without rehashing? Clearly state any assumptions.

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

(ii) $O(n)$

(c) (6 points) Suppose you use open addressing with hash functions $h_1(k) = 2k \mod N$ and $h_2(k) = 1 + (k \mod (N - 1))$. Illustrate the result of inserting the keys above using double hashing, i.e., $h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod N$.

```
H:  0  1  2  3  4  5  6  7  8
    5 15  6  7 16 17
```

(d) (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 with rehashing using a maximum load factor of $\frac{3}{4}$? Clearly state any assumptions.

(i) $O(n)$

(ii) $O(n^2)$
3. (20 points) **Binary Search Trees (BSTs).**

(a) (5 points) Draw the BST that would result from inserting the keys (6, 17, 23, 46, 3, 11) (in this order) into an initially empty BST. Show explicitly any intermediate restructuring that is required.

![BST](image)

(b) (5 points) Draw the resulting BST when the item with key 4 is removed form the BST shown below.

![BST](image) ⇒

(c) (5 points) Draw the AVL tree that would result from inserting the keys (6, 17, 23, 46, 3, 11) (in this order) into an initially empty AVL tree. Show explicitly any intermediate restructuring that is required.

![AVL](image) ⇒

(d) (5 points) Draw the resulting AVL when the item with key 3 is removed form the AVL shown below.

![AVL](image) ⇒
4. (20 points) **Priority Queues.** Show how to implement the Stack ADT using only a minimum priority queue and at most one additional data member. Your approach must work for any combination of \( k \) push\((e)\), pop\(()\), and top\(()\) operations. You do not need to justify complexity bounds, simply state the tightest bound.

(a) (5 points) **General approach.** Describe the general approach, assumptions, and extra data. Loosely explain why your approach will work.

My approach to ensure the FILO ordering that a stack produces is to use the negation of the size of the priority queue as the element’s key. (specifically, the size at item insertion). This ensures that the first element has the greatest key, and is thus the last one removed, and so forth. I will use no extra data members. Let \( PQ \) be the underlying priority queue.

(b) (5 points) **Push.** Describe your algorithm for push in pseudocode. What is the time complexity of this operation if the priority queue is implemented using an array-based heap?

**Algorithm** push\((e)\)

<table>
<thead>
<tr>
<th>Input: Element ( e )</th>
<th>( O(\log n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( PQ.\text{insert}(-PQ.\text{size}(), e) )</td>
<td></td>
</tr>
</tbody>
</table>

(c) (5 points) **Pop.** Describe your algorithm for pop in pseudocode. What is the time complexity of this operation if the priority queue is implemented with a sorted list?

**Algorithm** pop\(()\)

<table>
<thead>
<tr>
<th></th>
<th>( O(1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( PQ.\text{removeMin}() )</td>
<td></td>
</tr>
</tbody>
</table>

(d) (5 points) **Top.** Describe your algorithm for top in pseudocode. What is the time complexity of this operation if the priority queue is implemented with an unsorted list?

**Algorithm** top\(()\)

<table>
<thead>
<tr>
<th></th>
<th>( O(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( PQ.\text{min}() )</td>
<td></td>
</tr>
</tbody>
</table>
5. (20 points) Maps.

(a) (10 points) **Algorithm design.** Maps are used at various stages of compression and decompression algorithms. Often, these algorithms use symbol (e.g., a character in a string) frequencies in their more complicated routines. Design an algorithm using the Map ADT to compute all symbol frequencies in a large string $S$. Provide pseudocode and a description of your algorithm.

**Algorithm** `FREQUENCIES(S)`

**Input:** String $S$

**Output:** Frequency map of characters to integers

1: Map of characters to integers $M \leftarrow \emptyset$
2: for all Symbols $s \in S$ do
3: if $M$.get($s$) = $\emptyset$ then
4: $M$.put($s$, 1)
5: else
6: $M$.put($s$, $M$.get($s$) + 1)
7: return $M$

Essentially, my map contains characters as the key and counts as their values. For each character of the string, I add it to the map. If it already exists, the count is incremented by 1.

(b) (10 points) **Algorithm analysis.** A substitution cipher is a string encoding mechanism which replaces all instances of a character in the string with a different symbol from a given alphabet (set of symbols). Given the following algorithm for encoding:

**Algorithm** `ENCODE(S, M)`

**Input:** String $S$, Substitution map $M$

**Output:** Encoded string $S'$

1: $S' \leftarrow \emptyset$
2: for all Symbols $s \in S$ do
3: $S' \leftarrow S' \cup M$.get($s$)
4: return $S'$

Determine the complexity of this algorithm with the following implementations. Let $n$ be the number of symbols in the string $S$ and $m$ be the number of symbols in the alphabet. You do not need to justify complexity bounds, simply state the tightest bound.

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>i. Unsorted list</td>
<td>$O(nm)$</td>
</tr>
<tr>
<td>ii. Sorted table</td>
<td>$O(n \log m)$</td>
</tr>
<tr>
<td>iii. Direct address table</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>iv. Hash table with rehashing at a maximum load factor of $\frac{5}{8}$ assuming uniform hashing</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>v. AVL tree</td>
<td>$O(n \log m)$</td>
</tr>
</tbody>
</table>
6. **(Bonus 10 points)** You are given two lists of points $P, Q$ and a constant time function $\delta$ that takes two points and returns their distance in some metric space. Give an efficient algorithm for finding the $k$-closest pairs of points, where one point is in $P$ and the other in $Q$. Determine its complexity assuming the best implementation.

**Algorithm** \textsc{closest-pairs}(P, Q, k)

**Input:** List of points $P, Q$ and integer $k$

**Output:** $k$-closest pairs of points between $P$ and $Q$

1: Maximum priority queue of distance, pair entries $PQ \leftarrow \emptyset$
2: for all $p \in P$ do
3: \hspace{1em} for all $q \in Q$ do
4: \hspace{2em} Distance $d \leftarrow \delta(p, q)$
5: \hspace{2em} if $PQ.\text{size}() < k \lor d < PQ.\text{max}.\text{getValue}()$ then
6: \hspace{3em} $PQ.\text{insert}(d, \{p, q\})$
7: \hspace{2em} if $PQ.\text{size}() > k$ then
8: \hspace{3em} $PQ.\text{removeMin}()$
9: List of point pairs $L \leftarrow \emptyset$
10: while $\neg PQ.\text{isEmpty}()$ do
11: \hspace{1em} $L.\text{addFirst}(PQ.\text{removeMin}.\text{getValue}())$
12: return $L$

The complexity using an array-based heap is $O(|P||Q| \log k + k \log k)$. Here, there are $|P||Q|$ pairs total, and each pair could cause a remove or insert of the priority queue in the worst case. The last component is caused by $k$ removals from the priority queue.