Weight: 40 points (+5)

Directions: Do not discuss this exam with those who have not taken it. You have three hours to complete this exam. The exam is open book, open notes, in paper form only. You may not use any electronic devices or information provided by other students. Do not open the exam until instructed to do so.

In your answers, you may directly refer to the lectures, textbooks, homework solutions, or readers. Please write your answers on the exam itself, using the back of each page if necessary. You may use any standard library routines in your code except where prohibited by the problem statement. You are advised to read the entire exam and begin with questions that you find to be easy.

Do not write down any extraneous, impertinent, or incorrect information – you may be penalized if you do. In addition, for some questions on this exam, your solutions will be judged on performance, design, and style in addition to functionality.

Be sure that your name and login are on every sheet of this exam.

Your Name: ____________________________

Your Login: __________________________

Your TA’s Name: _________________________

1. _______ / 8
2. _______ / 6
3. _______ / 5
4. _______ / 8
5. _______ / 4
6. _______ / 6
7. _______ / 8

TOT _______ / 45
Throughout this exam, we will use the following conventions:

- Hash tables use chaining by adding items to the front of a linked-list at a given bucket
- Binary Search Trees (BSTs) are trees with a BST ordering property, a data field, and two BST children
- AVL-Trees are BSTs that satisfy and maintain the AVL-Tree property
- Singly-Linked Skip Lists are Skip Lists that use singly-linked lists instead of doubly-linked lists
- A Hamiltonian Cycle in a graph is a path through the graph that starts and ends at the same node and includes all other nodes exactly once
- Base 9 numbers are numbers represented by a sequence of digits, with digits ranging from 0 to 8

```java
/* Represents a TList */
class TListNode {
    static final int TLISTNODE = 0, ATOM = 1;

    TListNode data;
    TListNode next;

    /* Constructs a TListNode. All keys in next must be greater than
    * keys in data. */
    TListNode(TListNode data, TListNode next) {
        this.data = data;
        this.next = next;
    }

    /* Returns the type of this node */
    int type() {
        return TLISTNODE;
    }
}

/* Represents an Atomic TList item (a key) */
class Atom extends TListNode {
    int key;

    /* Constructs an atomic TListNode that represents the given key.
    * All keys in next must be greater than the given key. */
    Atom(int key, TListNode next) {
        super(null, next);
        this.key = key;
    }

    /* Returns the type of this node */
    int type() {
        return ATOM;
    }
}
```
1. (9 Fingers) Among archaeological finds at a recent dig site were notes on a mathematics system used by an ancient human race. You are asked to help understand these notes which may help researchers understand their civilization. You find, to your dismay, that members of this ancient race had only 9 fingers and 9 toes, leading them to use a base 9 number system.

To aid in your translation, you need to write code to convert from base 10 to base 9 numbers, but since you will be handling large numbers, you would like to optimize for space. Fill in the methods below to match the following descriptions:

The constructor for the class Base9 constructs a base 9 representation of the given decimal long value. Each base 9 digit is stored in the digits array of byte values, but to conserve space, the first base 9 digit is stored in the least-significant 4 bits of the first byte, the second digit is stored in the most-significant 4 bits of the first byte, the third digit is stored in the least-significant 4 bits of the second byte, and so on. Lower array indicies hold less significant digits of the value. You may assume that all values are non-negative. In addition, you will need a way to indicate that there are no more digits in the array (in the case of numbers with odd numbers of digits).

Afterwards, fill in the method add that adds a Base9 value to the current Base9 value, yielding a new Base9 value. The method add should be non-destructive – the two original Base9 values should not be modified in any way as a result of the call.

In your solutions for this question and question 2, you may use only primitives, arrays of primitives, instances of Base9 objects, and arrays of Base9 objects. Solutions that violate this requirement will not be graded. You may use any operators that operate on primitive values.

```java
class Base9 {
    byte[] digits;

    /* Constructs a Base9 number representation of the given decimal value. */
    Base9(long value) {
        // FILL IN
    }
}
```

Continued on next page.
/* Returns the sum of this Base9 number and the given Base9 number. *
* Does not modify the argument or this. */
Base9 add(Base9 value) {
    // FILL IN

}
2. **(Radix Sort, Base 9)** Fill in the following method that performs a least-significant-digit first radix sort of the given input Base9 value array. Your solution may not use any predefined sorting routines, may not modify the input array, and may only use primitives, arrays of primitives, Base9 objects, and arrays of Base9 objects. Furthermore, you **may not** assume that the given Base9 values all have the same number of digits. You may assume, however, that your constructor for the class Base9 works as specified.

```java
static Base9[] radixSort(Base9[] values) {
    // FILL IN
}
```

// FILL IN ADDITIONAL METHODS IF NEEDED
3. **(Synthesis)** Answer the questions below with a short paragraph for each.

a. Suppose a new scheduling algorithm, LET, is used to schedule processes in a system. LET, which stands for Longest Elapsed Time, runs almost like SET (Shortest Elapsed Time), except it always chooses the job that has run for the most time (as opposed to the least time as in SET). We can easily implement LET by replacing the minimum priority queue we used for SET with a maximum priority queue. Assume that LET uses quantums of length $q$ seconds. Evaluate LET with respect to its average turnaround time and average wait time for processes. Recall that the turnaround time for a process is the sum of its wait time and run time. How does LET compare to SET given these metrics? Be precise and provide an example or two.

b. Suppose a greedy algorithm is used to find a maximum weight Hamiltonian cycle in a weighted graph given a starting vertex. The algorithm chooses the maximum weight edge from the current vertex at each step. It is found that the algorithm fails on certain input graphs. Describe why this algorithm and other greedy algorithms could fail and give an example graph for which this algorithm fails.
4. **(Selection)** Answer the questions below by circling the correct answers. If more than one answer is correct, circle all correct answers. If no answers are correct, circle None. Circle answers clearly.

a. Which of the following run in O(1) time on a hash table (where \( k_i \) is a key and \( v_j \) is a value), assuming a good hash code and a load factor of 2?

\[
\text{Insert}(k_0, v_0) \quad \text{FindKey}(k_0) \quad \text{FindValue}(v_0) \quad \text{Remove}(k_0) \quad \text{RangeQuery} \quad (k_0, k_1) \quad \text{None}
\]

b. Which of the following run in O(1) time on a hash table (where \( k_i \) is a key and \( v_j \) is a value), assuming an arbitrarily bad hash code and a load factor of 1.5?

\[
\text{Insert}(k_0, v_0) \quad \text{FindKey}(k_0) \quad \text{FindValue}(v_0) \quad \text{Remove}(k_0) \quad \text{RangeQuery} \quad (k_0, k_1) \quad \text{None}
\]

c. An order 20 B-Tree of height 3 (3 non-empty levels) contains \( n \) keys. \( n \) could be:

\[
1 \quad 10 \cdot 9 \quad 2 \cdot 10 \cdot 9 \quad 1 + (2 \cdot 9) + (2 \cdot 9^2) \quad 2 \cdot 20 \cdot 19 \quad 20^3 - 1 \quad 20^3 \quad \text{None}
\]

d. An order 20 B+ Tree of height 3 (3 non-empty levels) contains \( n \) data items. \( n \) could be:

\[
10 \cdot 10 \quad 2 \cdot 9 \cdot 9 \quad 2 \cdot 10 \cdot 9 \quad 10 \cdot 10 \cdot 9 \quad 10^3 - 1 \quad 20 \cdot 20 \cdot 19 \quad 20^3 - 1 \quad 20^3 \quad \text{None}
\]

e. Which of the following uses less space (fewer pointers) to store \( n \) items than an AVL-Tree?

- Singly-Linked List
- Array
- Doubly-Linked List
- Singly-Linked Skip List
- Vector
- Binary Search Tree
- None

f. If insertion into a skip list of \( n \) items were changed such that items are placed at upper levels with probability 0.9, a find operation would take:

\[\Theta(1) \quad \Theta(lg n) \quad \Theta((lg n)^2) \quad \Theta(n) \quad \Theta(n \ lg \ n) \quad \Theta(n^2) \quad \text{None}\]

g. If a binary heap of \( n, n \geq 1 \) elements has another item inserted into it, then for a fixed value \( n \), the number of swap operations to restore the heap property could be:

\[
0 \quad 1 \quad 2 \quad [\lg(n + 1)] \quad [\lg(n + 1)] \quad [\lg(n + 1)] + 1 \quad \frac{n}{2} \quad n - 1 \quad \text{None}
\]

h. I’m moving home with my drums and CS books, but my car doesn’t have room for them separately. I decide to pack my books inside my drums (I actually have done this before) – if I have \( n \) books, how long will it take me to figure out an space-optimal way to pack my books, assuming I solve this computationally with the fastest algorithm I know?

\[\Theta(1) \quad \Theta(lg n) \quad \Theta(n^2) \quad O(n^3) \quad \Omega(n^3) \quad \Omega(2^n) \quad \Omega(n^n) \quad \text{None}\]
5. (City Roads) You are hired to replace the transportation manager for a large city – it seems that the old transportation manager had turned far too many streets into one-way streets, leaving some residents without a way of getting from one place to another. Looking at the city road map, you decide that an easy solution is to turn some of the one-way streets into two-way streets. You will need to turn enough one-way streets into two-way streets so that drivers can get from any place in the city to any other place in the city.

Describe how to represent this problem computationally (using data structures). Describe an algorithm that selects the one-way streets that must be turned into two-way streets as required above such that the length of road you convert from one-way to two-way is minimized. Be precise (use pseudocode when necessary).
6. (TList) Fill in the method `cut` below such that it selects elements within a range from the given `TListNode` object structure (which represents what we will call a TList) and returns a pointer to the first `TListNode` of the resulting TList. Your implementation may not use `new` and must run in $O(t)$, where $t$ is the number of elements in the TList. Read the definitions for `TListNode` and `Atom` (given on page 2) very carefully.

```cpp
/* Returns the first TListNode in the TList that contains all elements (keys) in the argument TList that are greater than L and less than H */
TListNode cut(TListNode tlist, int L, int H) {
    // FILL IN
}

// FILL IN ADDITIONAL METHODS IF NEEDED
```
7. (Design) In this question you are to design data structures and algorithms to solve problems. Your solutions will be evaluated on functionality, performance (time and space), elegance, and above all, precision. Your solutions for each question should be cohesive – the parts are interdependent.

a. In an effort to catch drivers that break the speed limit, the DMV joins forces with police departments statewide to create BigBrother devices. BigBrother devices can be set in any location by police to monitor traffic on the street. The devices capture the license plate numbers and speeds of all passing cars, storing them with the location at which the information was recorded.

At the end of the month, all BigBrother devices are gathered and all their data transferred to a central machine that will then issue tickets. This central machine has access to a listing (a hash table) of locations mapped to speed limits. In total, all BigBrother devices statewide record n entries a month, at a total of c locations, and of the n entries, k were speeding. You may assume that c < k < n.

1. Describe how data should be stored within each BigBrother device. Remember that the devices can be moved to different locations within a month.

2. Describe how the data from each device should be transferred and stored in the central machine.

3. Describe an algorithm that runs on the central machine in O(k) time that produces a list of all speed violations.
The US Postal Service decides to bring itself into the 21st century by offering permanent email addresses for all US citizens. However, since it hasn’t upgraded its network infrastructure in years, the designers of this system decide to distribute users’ email regionally. The country is divided into regions, each of which are themselves divided into sub-regions, and so on. At the most local level, users’ email data is stored. Although email will be accessible from anywhere, since most accesses will be from nearby, this distribution will decrease the load on the USPS network.

When email is sent from one of these accounts to another, it will travel over the USPS network, but for efficiency reasons, the designers want it to travel over the fewest network links possible. Furthermore, since users will move reasonably often, their data should be easy to remove from one regional data center and add to another.

1. Describe a data structure for regional data centers (at all levels) to store information about users and the locations of their data. Describe how a search could be performed from anywhere for any user’s data.

2. Describe a way to route email through data centers such that the email stays as local as possible.

3. Describe how, at the local level, multiple users’ email could be stored across a number of different server computers that together would act as the local “data center”. Describe how searches for user email would be performed.