1. - How exactly is the skip-list constructed from layers of linked-lists? What does each of the linked-lists constituting the skip-list store?

A： Skip-List consists of multiple layers of ordered linked lists. The bottom layer of linked lists contains all elements, and each upper layer of linked lists is a subset of the lower layer, with the element spacing increasing gradually. Each layer of linked lists stores references to some elements, and the elements in the higher layer of linked lists serve as "jumping points" to speed up the search.

2. What is the height of a skip-list? How can we determine the height when adding new elements?

A：

The height is the maximum number of layers in the Skip-List.

When inserting an element, the number of layers is determined by "flipping a coin". For example, if the coin flip sequence is HHT, the number of layers is 3.

3. What is the runtime complexity for searching, insertion, and removal operations on the skip-list? What does it mean when we say that the complexity of these operations is '***expected***'? What is the difference between an ***expected* bound** on runtime complexity and a ***worst-case* bound**?

A：

The expected time complexity of searching, inserting, and deleting is O(log n).

"Expected" refers to the performance in the average case, which depends on randomness; the worst case may be O(n), that is, all elements are concentrated in the same layer.

The expected bound is the average probability, and the worst case bound considers all possible inputs.

4. Compare the skip-list to other data-structures that you already know, e.g., a list collection, singly and doubly linked lists. What are the advantages and disadvantages of the skip-list? Compare the complexities of the basic operations offered by the skip-list to those of the mentioned data structures.

A：

Data structure	Advantages	Disadvantages	Complexity
Skip-List	O(log n) expected time/ Simple to use	Additional space overhead	O(log n)
Singly linked list	Simple, low memory	Linear time operation	O(n)

Balanced tree	O(log n) worst-case time	Hard to achieve	O(log n)

5. How exactly does each of the standard operations work? Here, you should be ready to explain the sequence of actions required to perform searching, insertion, and removal from the skip-list.

A：

Search: Start from the highest level, go right to find the largest node that is smaller than the target, sink to the next level, and continue until the bottom level.

Insert: Determine the number of new node levels, insert each level, and update the pointer.

Delete: Find the position of the node in each level, delete each level, and repair the pointer.

2. Solve the following numeric example. Given the sequence of ***coin tosses***, where H stands for ***head*** and T stands for ***tails***:

T H T T H H T T H T H T H H T H H T T H H T H T T H T T T H T H T H T T H T H H T T.

Build a skip-list containing the following values:

1 40 11 85 86 5 0 8.

Show the full state of the skip-list after each insertion. Note that *you must explicitly state the meaning you attach* to *heads* and *tails* at the start of your answer. Remember that you must start tossing the coin from left side of the sequence and may not need to use all the tosses to build the skip-list. You may assume as the height resolution policy that we allow a tower to grow as long as heads (or tails) keep getting returned from the given sequence, which emulates a random number generator.

A： Based on the insertion order: 1 -> 40 -> 11 -> 85 -> 86 -> 5 -> 0 -> 8

Element	Coin	Number of Layers
1	T	1
40	H -> T	2
11	T	1
85	H-> H -> T	3
86	T	1
5	H -> T	2
0	H -> T	2
8	H-> H-> T	3

Insertion process:

1. Insert 1 (Layer 1)

Layer 1: 1

2. Insert 40 (Layer 2)
   Layer 2: 40
   Layer 1: 1 -> 40

3. Insert 11 (Layer 1)
   Layer 2: 40
   Layer 1: 1 -> 11 -> 40

4. Insert 85 (Layer 3)
   Layer 3: 85
   Layer 2: 40 -> 85
   Layer 1: 1 -> 11 -> 40 -> 85

5. Insert 86 (Layer 1)
   Layer 3: 85
   Layer 2: 40 -> 85
   Layer 1: 1 -> 11 -> 40 -> 85 -> 86

6. Insert 5 (Layer 2)
   Layer 3: 85
   Layer 2: 5 -> 40 -> 85
   Layer 1: 1 -> 5 -> 11 -> 40 -> 85 -> 86

7. Insert 0 (Layer 2)
   Layer 3: 85
   Layer 2: 0 -> 5 -> 40 -> 85
   Layer 1: 0 -> 1 -> 5 -> 11 -> 40 -> 85 -> 86

8. Insert 8 (Layer 3)
   Layer 3: 8 -> 85
   Layer 2: 0 -> 5 -> 8 -> 40 -> 85
   Layer 1: 0 -> 1 -> 5 -> 8 -> 11 -> 40 -> 85 -> 86

Final Skip-List status:

Layer 3:8 -> 85

Layer 2: 0 ->5 -> 8 -> 40 -> 85

Layer 1: 0 -> 1 -> 5 -> 8 -> 11 -> 40 -> 85 -> 86