1. Insert 10, 15, 8, 2, 23, 13 in that order into a hash table of size 10 using separate chaining and the hash function $h(x) = 3x + 7$.

Suppose the hash table uses dynamic resizing and is doubled in size when the load factor threshold reaches 0.9. Continue inserting the values 7, 11, 3, 24. (Notice how elements can be moved during the resize)

2. Why can’t a load factor threshold of 1.1 be chosen for a hash table when using linear probing?

3. In terms of $n$, what is the runtime for the following piece of code?

```cpp
std::vector<int> v;
for (int i = 0; i < n; i++)
    v.insert(v.begin(), i);
```

For this piece of code?

```cpp
std::vector<int> v;
for (int i = 0; i < n; i++)
    v.push_back(i);
```

What if the type of $v$ is changed to be a `std::list`?

4. As seen in class, an in-order traversal of a binary search tree will list the elements of the tree in sorted order in $O(n)$ time. How quickly can a comparison based binary search tree construction algorithm take?

5. Given a list of numbers, how can you create a binary rooted tree with that list as its in-order traversal? as its pre-order traversal? as its post-order traversal?