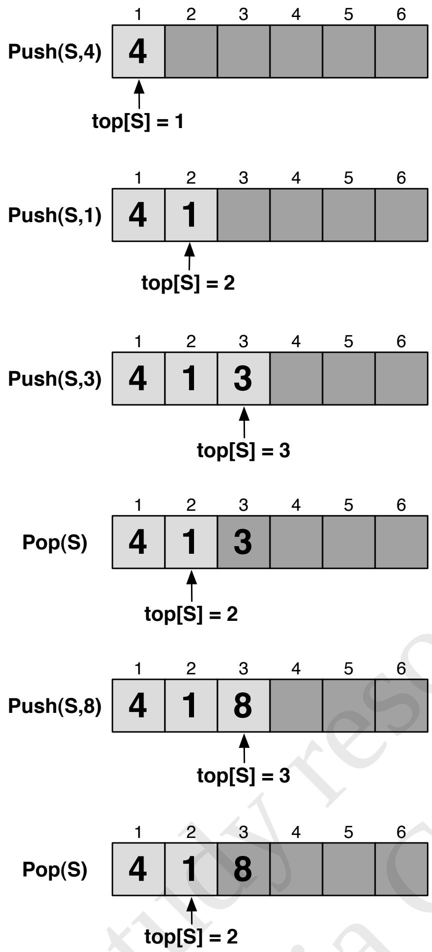
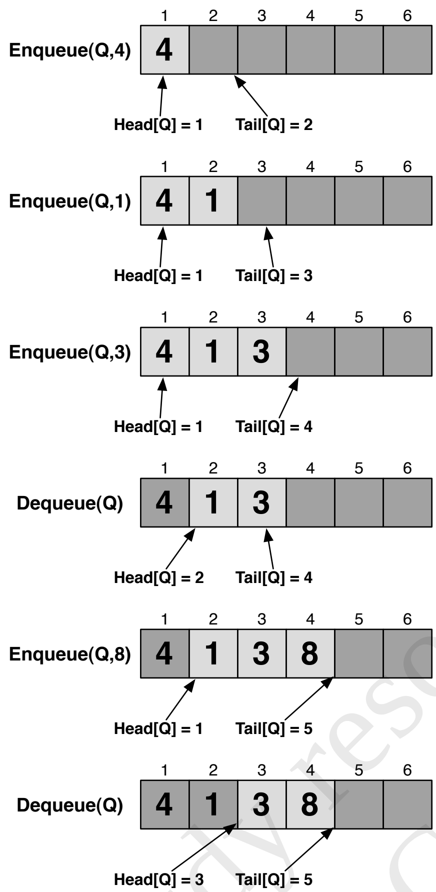
Exercise 10.1-1



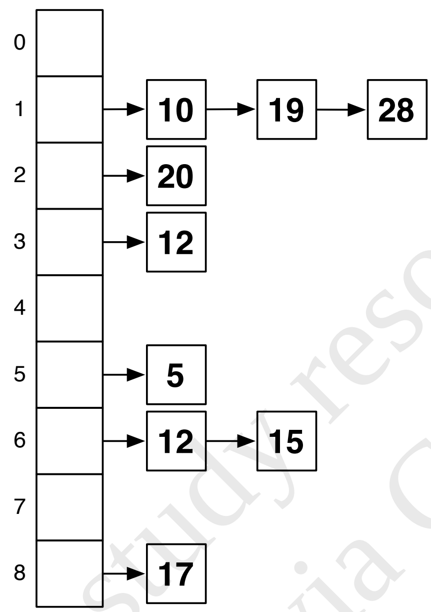
Exercise 10.1-3



Problem 10-1

	$Unsorted, Single$	$Sorted, Single$	$UnsortedDouble$	$SortedDouble$
$Search(L, k)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
$Insert(L, x)$	$O(1)$	$O(n)$	$O(1)$	$O(n)$
$Delete(L, x)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$
$Successor(L, x)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$
$Predecessor(L, x)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$
$Minimum(L)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$
$Maximum(L)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$

Exercise 11.2-2



Exercise 11-2-3
An average has should take around $1 + \alpha$ where $\alpha < 1$ to perform an operation. To see if we see an increase in performance we check each operation.

Successful Seach
$1 + \frac{\alpha}{2}$ for both

Unsuccessful Seach
$1 + \frac{\alpha}{2}$ for sorted
$1 + \alpha$ for unsorted

Insert
Assume that the key does not exist, so this is an unsuccessful search followed by a constant time insert.
$1 + \frac{\alpha}{2}$ for sorted
$1 + \alpha$ for unsorted

Delete
Assume key exists, so this is a successful search followed by a constant time delete.
$1 + \frac{\alpha}{2}$ for both

In the best case we would improve from $1 + \alpha$ to $1 + \frac{\alpha}{2}$ resulting in rather small performance gains.

Exercise 11.3-4
$h(61) = 700$
$h(62) = 318$
$h(63) = 936$
$h(64) = 554$
$h(65) = 172$

Problem 11-3a

Using the probe scheme we find that the probe sequence will be:

$$h(k), h(k)+1, h(k)+1+2, h(k)+1+2+3, \ldots$$

where all elements are modulo m.

Using this we see that the i^{th} offset in the probe sequence is:

$$\sum_{k=0}^{i} j = \frac{i(i+1)}{2} = \frac{i^2}{2} + \frac{i}{2} \tag{1}$$

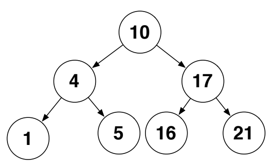
This gives us a probe sequence of:

$$h'(k,i) = \left(h(k) + \frac{i^2}{2} + \frac{i}{2}\right) \mod m$$

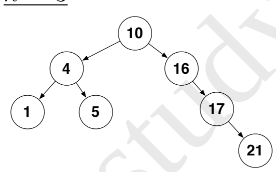
which is an instance of quadratic hashing.

Exercise 12-1.1

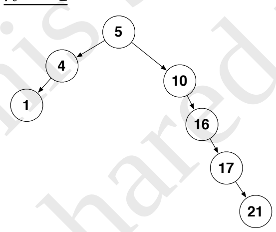
$h = 2$



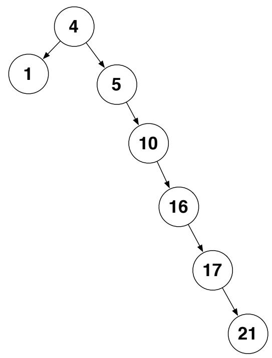
$h = 3$



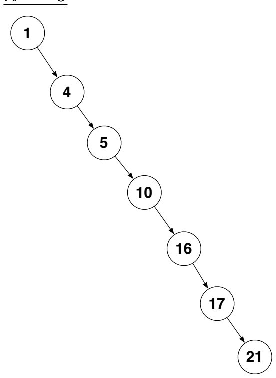
$h = 4$



$\underline{h=5}$



$\underline{h=6}$



Solution to Exercise 12.1-2

In a heap, a node's key is $\geq$ both of its children's keys. In a binary search tree, a node's key is $\geq$ its left child's key, but $\leq$ its right child's key.

The heap property, unlike the binary-searth-tree property, doesn't help print the nodes in sorted order because it doesn't tell which subtree of a node contains the element to print before that node. In a heap, the largest element smaller than the node could be in either subtree.

Note that if the heap property could be used to print the keys in sorted order in $O(n)$ time, we would have an $O(n)$-time algorithm for sorting, because building the heap takes only $O(n)$ time. But we know (Chapter 8) that a comparison sort must take $\Omega(n \lg n)$ time.

Exercise 12.1-5

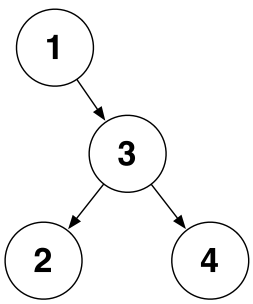
Assume that a BST could be constrcuted in time less than $\Omega(n \log n)$. Also, recall that an inorder tree traversal can be done on a BST in $O(n)$ time and this traversal will produce the elements in sorted order. This would mean that a list could be sorted in less than $\Omega(n \log n)$ tim by build a BST from the data and then performing an inorder tree traversal. This contradicts the assumption that any comparison based sort is $\Omega(n \log n)$.

Exercise 12.2-1

(c) and (e) are invalid, the rest are valid.

12.2-4

Four is the smallest possible counter example, for instance, take the following BST:



$A = \{2\}$
$B = \{1, 3, 4\}$
$C = \emptyset$

Let $a = 2, b = 1$, out claim $a \leq b \rightarrow 1 \leq 2$. Contradiction

Solution to Exercise 12.3-3

Here's the algorithm:

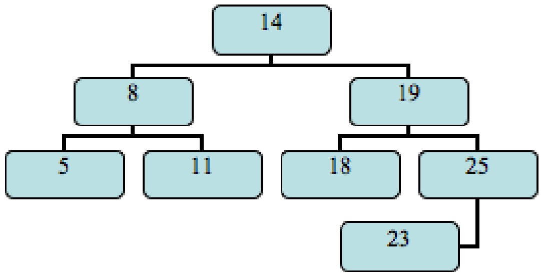
```
TREE-SORT(A)
let T be an empty binary search tree
for i ← 1 to n
     do TREE-INSERT(T, A[i])
INORDER-TREE-WALK(root[T])
```

Worst case: $\Theta(n^2)$—occurs when a linear chain of nodes results from the repeated TREE-INSERT operations.

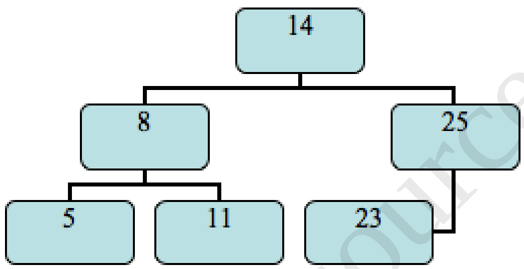
Best case: $\Theta(n \lg n)$—occurs when a binary tree of height $\Theta(\lg n)$ results from the repeated TREE-INSERT operations.

Exercise 12.3-5

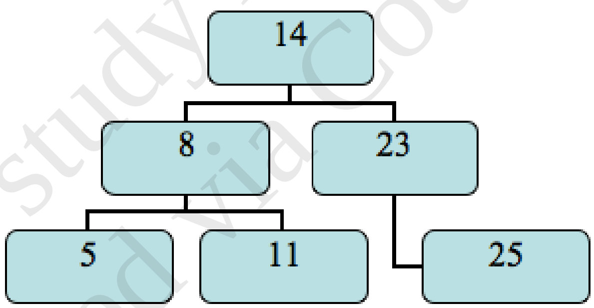
Deletion is not commutative, here is one counterexample



(a) BINARY SEARCH TREE



(b) BINARY SEARCH TREE AFTER DELETION OF 18 and then 19



(c)BINARY SEARCH TREE AFTER DELETION OF 19 and then 18