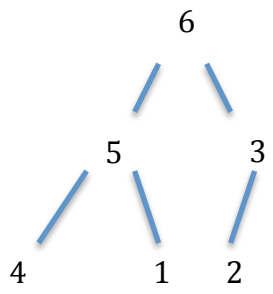
COMP 251 Assignment 1
Chelsea Ma
260515648

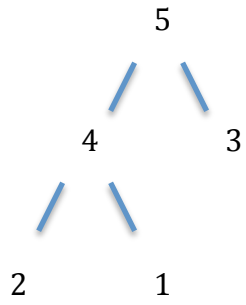
Question 2: Operation of Heapsort
Step 1: Max heap data structure after Build Max Heap

1	2	3	4	5	6
6	5	3	4	1	2



Step 2: Max heap data structure after first iteration of Max-Heapify

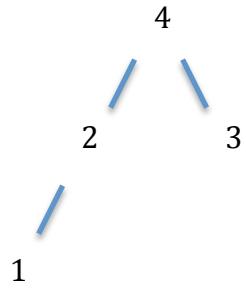
5	4	3	2	1



Discarded: 6

Step 3: Max heap data structure after second iteration of Max-Heapify

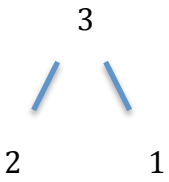
4	2	3	1



Discarded: 5, 6

Step 4: Max heap data structure after third iteration of Max-Heapify

3	2	1



Discarded: 4, 5, 6

Step 5: Max heap data structure after fourth iteration of Max-Heapify

2	1

Discarded: 3, 4, 5, 6

Step 6: Max heap data structure after fifth iteration of Max-Heapify

1

Discarded: 2, 3, 4, 5, 6

Final Result:

1	2	3	4	5	6

Question 3: Hash Function Collision

Load factor, $\propto = n/m$, represents the average number of elements stored in a chain.
For each pair of keys a, b where $a \neq b$, $Pr\{h(a)=h(b)\} = 1/m$ because hash values are chosen randomly and independently from the array. Expected value of indicator random variable $E[X_{ab}] = 1/m$.
Let Y be the total number of collisions so that $Y = \Sigma X_{ab}$.
Thus, expected number of collisions is

$E[Y] = E[\Sigma X_{ab}]$
$= \Sigma E[X_{ab}]$
$= \binom{n}{2} 1/m$

= (n(n-1))/2 * 1/m
= (n(n-1))/2m
= (n/m) * (n-1)/2
$\geq \frac{n}{m}$ if n≥3
$\geq \alpha$ if n≥3

Thus, expected number of collision is greater than or equal to the load factor if n≥3.
The big oh is n^2 .

Question 4: Binary Tree

```
RotateLeft(B, x) {
        if right[x] and left [x] ≠ nil
        father[x] = right[x]  //right child of x is now parent of x
        right[x] = left[(father[x])]   //left child of right child is now right child of x
        left[(father[x])]  = x  //x is now left child of parent
        return father[x]
        else
        return x  //if x is a leaf we can't rotate anything so return x
}
```

Question 5: Binary Tree

Let T(n) be number of binary search tree with n nodes.
We consider all possible binary search tree with each key at the root.
If there are n keys, for each n choices of the key at the root, there are n-1 non-root nodes that are partitioned into two that are less than the key of the root and that are greater than the key of the root.
Let i be the key of the root.
Then there are i-1 keys smaller than i and n-i keys greater than i.
With i-th at root, T(n) = T(i-1) * T(n-i) because left and right subtrees are independent.
Base cases are T(0) = 1 (when tree is empty) and T(1) = 1 (when tree has one node).

Question 1:
For a small number of inputs, I would say that multiplication method is better because I get less number of collisions. However, the number of inputs increase division method seems to be better because it produces less number of collisions.