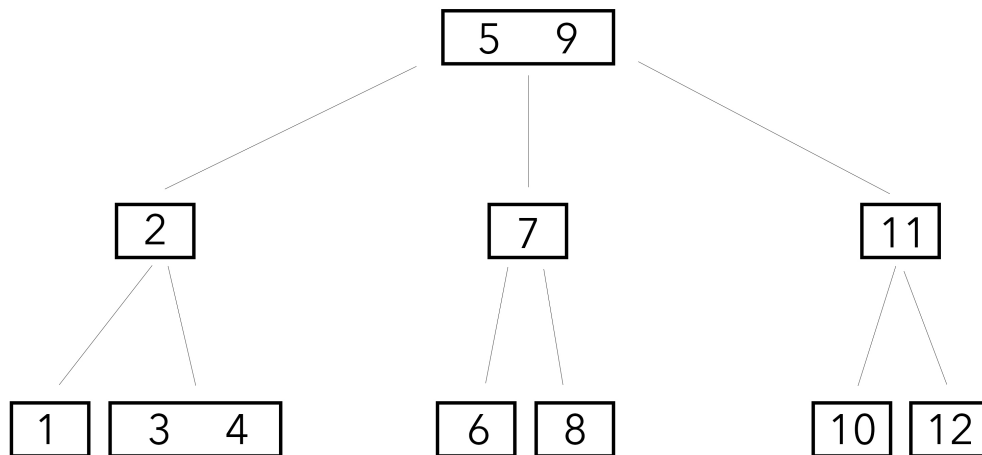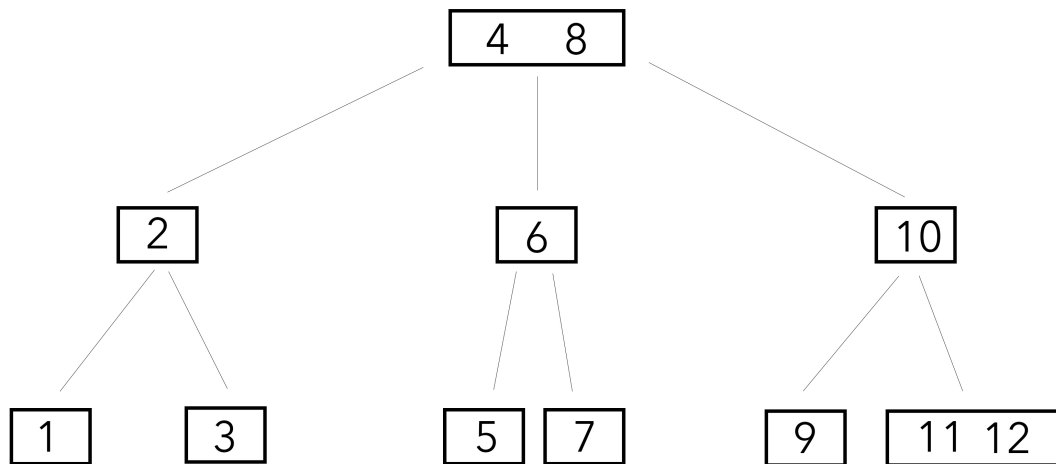Eric Ong (861267345)
Thomas Chin (861290572)
Section 002

CS014 Homework #3

1. **A) Insert the following 12 values into an initially empty 2-3 tree (insert them in this exact order): 3,5,7,1,2,4,6,10,9,8,12,11. You only need to show the tree after all the insertions, not the tree after each insertion.**

```
                        ┌─────────┐
                        │  5   9  │
                        └─────────┘
              ┌───────────┬─────────────┐
          ┌──────┐    ┌──────┐      ┌──────┐
          │  2   │    │  7   │      │  11  │
          └──────┘    └──────┘      └──────┘
          ┌───┐ ┌───────┐  ┌───┐ ┌───┐  ┌────┐ ┌────┐
          │ 1 │ │ 3   4 │  │ 6 │ │ 8 │  │ 10 │ │ 12 │
          └───┘ └───────┘  └───┘ └───┘  └────┘ └────┘
```

B) Insert the same 12 values into an initially empty 2-3 tree, only insert them in ascending order this time: 1,2,3,4,5,6,7,8,9,10,11,12. Again, you only need to show the tree after all insertions.
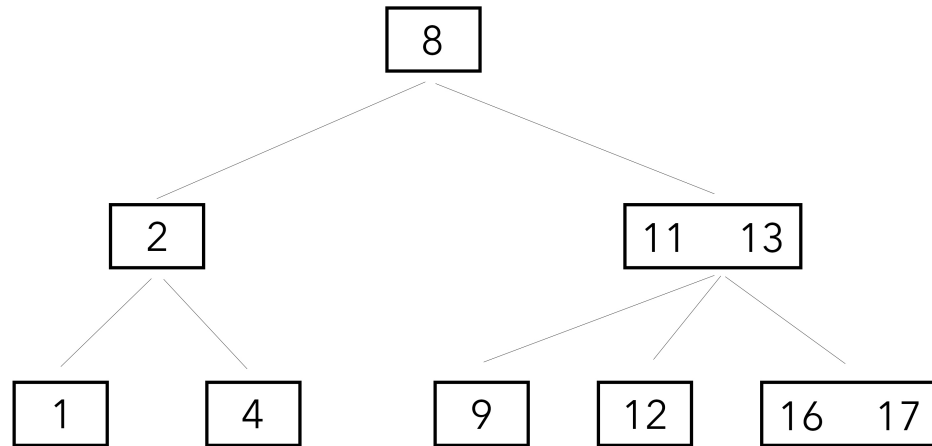
```
                        ┌─────────┐
                        │  4    8  │
                        └─────────┘
              ┌──────────────┼──────────────┐
          ┌─────┐         ┌─────┐         ┌─────┐
          │  2  │         │  6  │         │ 10  │
          └─────┘         └─────┘         └─────┘
           ┌───┴───┐       ┌───┴───┐       ┌───┴────┐
        ┌─────┐ ┌─────┐ ┌─────┐ ┌─────┐ ┌─────┐ ┌───────┐
        │  1  │ │  3  │ │  5  │ │  7  │ │  9  │ │ 11  12 │
        └─────┘ └─────┘ └─────┘ └─────┘ └─────┘ └───────┘
```

C) Finally, insert these same 12 values into a BST in ascending order (in the same order you inserted them into the 2-3 tree in part b): 1,2,3,4,5,6,7,8,9,10,11,12.

```
1
 2
  3
   4
    5
     6
      7
       8
        9
         10
          11
           12
```
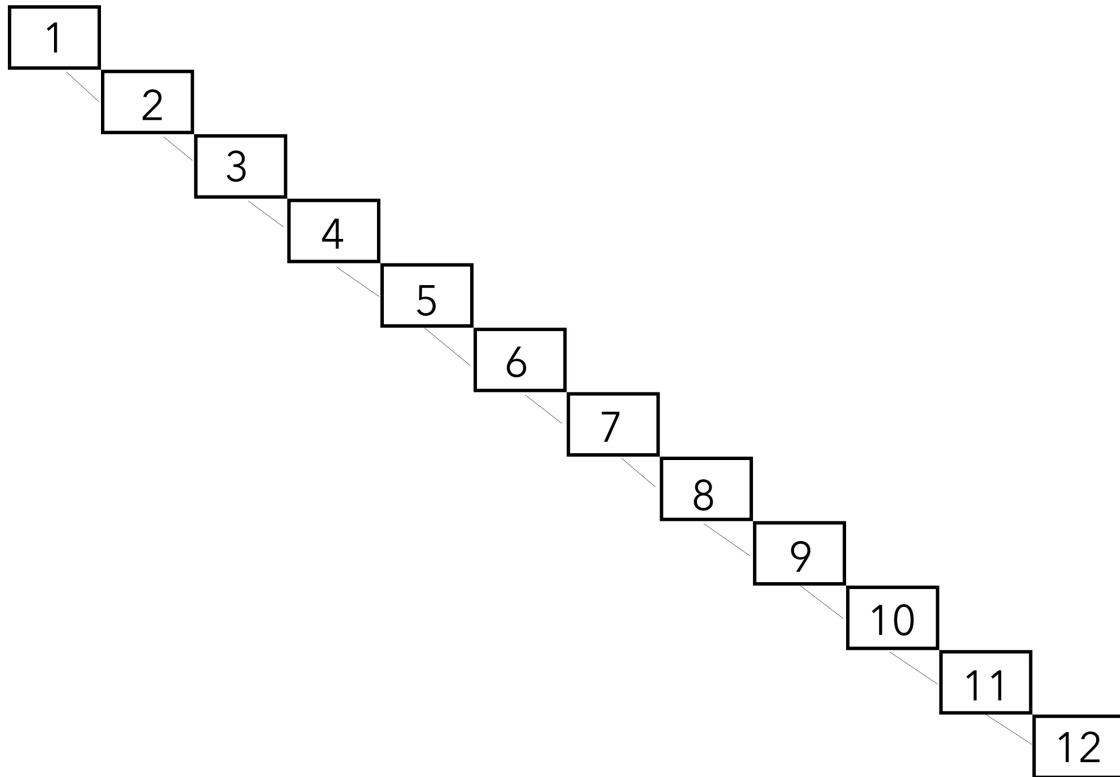
D) How many key comparisons does it take to NOT find the value 13 within the 2-3 tree in part b? How many key comparisons does it take to NOT find the value 13 within the BST in part c?

2-3 Tree has 3 comparisons, BST has 12 comparisons

2. Given the following 2-3 tree show the tree after removing
   the values 14, 10 & 15, in this order. Just show the tree
   after all 3 removals, not after each removal.

```
                            ┌───┐
                            │ 8 │
                            └───┘
                 ┌────────────┘  └────────────┐
              ┌───┐                      ┌────────┐
              │ 2 │                      │ 11  13 │
              └───┘                      └────────┘
            ┌──┘ └──┐            ┌─────────┘ │ └─────────┐
         ┌───┐    ┌───┐      ┌───┐       ┌────┐      ┌────────┐
         │ 1 │    │ 4 │      │ 9 │       │ 12 │      │ 16  17 │
         └───┘    └───┘      └───┘       └────┘      └────────┘
```

3. **An empty table has a capacity of 100, and you insert 6 entries with keys 100,0,119,1299,1399, and 2. Using linear probing and the basic has function (key mod size), where will these entries be placed in the table? Where will they be placed with double hashing (with HF2 = 1 + (key mod 98))?**

HF = key % size, F(i) = i
100: ((100 % 100) + 0) % 100 = **0**
0: ((0 % 100) + 0) % 100 = **0, COLLISION**
   ((0 % 100) + 1) % 100 = **1, PROBE**
1199: ((1199 % 100) + 0) % 100 = **99**
1299: ((1299 % 100) +0) % 100 = **99, COLLISION**
     ((1299 % 100) +1) % 100 = **0, PROBE**
     ((1199 % 100) +2) % 100 = **1, PROBE**
     ((1199 % 100) +3) % 100 = **2, PROBE**
1399: ((1399 % 100) + 0) % 100 = **99, COLLISION**
     ((1399 % 100) + 1) % 100 = **0, PROBE**
     ((1399 % 100) + 2) % 100 = **1, PROBE**
     ((1399 % 100) + 3) % 100 = **2, PROBE**
     ((1399 % 100) + 4) % 100 = **3, PROBE**
2: ((2 % 100) + 0) % 100 = **2, COLLISION**
   ((2 % 100) + 1) % 100 = **3, PROBE**
   ((2 % 100) + 2) % 100 = **4, PROBE**

HF2 = 1 + (key % 98)
100: ((1+(100 % 98)) + 0) % 100 = **3**
0: ((1 + (0 % 98)) + 0) % 100 = **1**
1199: ((1 + (1199 % 98)) + 0) % 100 = **24**
1299: ((1 + (1299 % 98)) + 0) % 100 = **26**
1399: ((1 + (1399 % 98)) + 0) % 100 = **28**
2: ((1 + (2 % 98))+ 0) % 100 = **3, COLLISION**
   ((1 +(2 % 98))+ 1) % 100 = **4, PROBE**

4. Use a has table with size of 811 that uses separate
   chaining to specify where the following entries will be
   inserted. Use a basic has function (key mod size). Insert
   811,0,1623,2435,3247, and 2.

   811: 811 % 811 = **0**
   0: 0 % 811 = **0**
   1623: 1623 % 811 = **1**
   2435: 2435 % 811 = **2**
   3247: 3247 % 811 = **3**
   2: 2 % 811 = **2**

5. **Use a hash table of size 11 and the basic hash function (key mod size). Insert the following items using quadratic probing: 2,14,21,36,70,18,47,32, and 11.**

HF = key % size, F(i) = i^2
2: ((2 % 11) + 0^2) % 11 = **2**
14: ((14 % 11) + 0^2) % 11 = **3**
21: ((21 % 11) + 0^2) % 11 = **10**
36: ((36 % 11) + 0^2) % 11 = **3, COLLISION**
     ((36 % 11) + 1^2) % 11 = **4, PROBE**
70: ((70 % 11) + 0^2) % 11 = **4, COLLISION**
     ((70 % 11) + 1^2) % 11 = **5, PROBE**
18: ((18 % 11) + 0^2) % 11 = **7**
47: ((47 % 11) + 0^2) % 11 = **3, COLLISION**
     ((47 % 11) + 1^2) % 11 = **4, PROBE**
     ((47 % 11) + 2^2) % 11 = **7, PROBE**
     ((47 % 11) + 3^2) % 11 = **1, PROBE**
32: ((32 % 11) + 0^2) % 11 = **10, COLLISION**
     ((32 % 11) + 1^2) % 11 = **0. PROBE**
     ((32 % 11) + 2^2) % 11 = **3, PROBE**
     ((32 % 11) + 3^2) % 11 = **8, PROBE**
11: ((11 % 11) + 0^2) % 11 = **0, COLLISION**
     ((11 % 11) + 1^2) % 11 = **1, PROBE**
     ((11 % 11) + 2^2) % 11 = **4, PROBE**
     ((11 % 11) + 3^2) % 11 = **9, PROBE**

6. **Rehash the resulting table from the previous question to a table size of 23.**

```
2: ((2 % 23) + 0^2) % 23 = 2
14: ((14 % 23) + 0^2) % 23 = 14
21: ((21 % 23) + 0^2) % 23 = 21
36: ((36 % 23) + 0^2) % 23 = 13
70: ((70 % 23) + 0^2) % 23 = 1
18: ((18 % 23) + 0^2) % 23 =  18
47: ((47 % 23) + 0^2) % 23 = 1, COLLISION
      ((47 % 23) + 1^2) % 23 = 2, PROBE
      ((47 % 23) + 2^2) % 23 = 5, PROBE
32: ((32 % 23) + 0^2) % 23 = 9
11: ((11 % 23) + 0^2) % 23 = 11
```