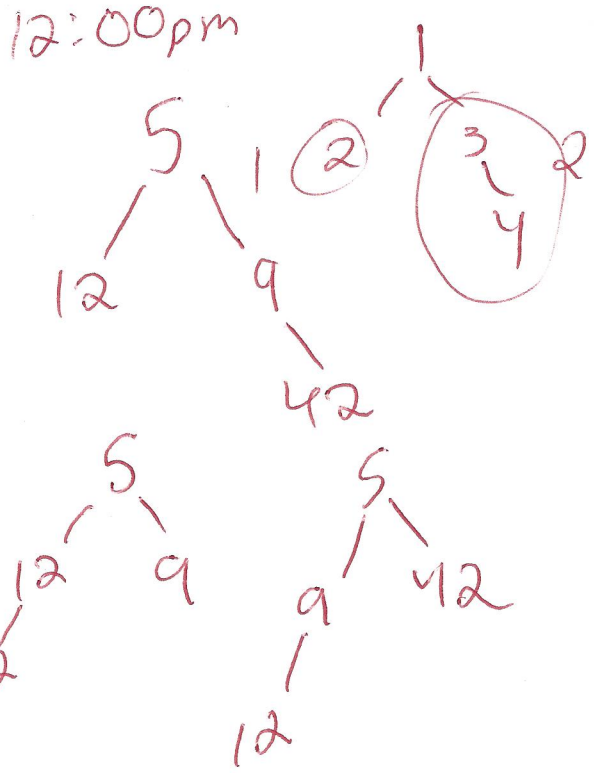
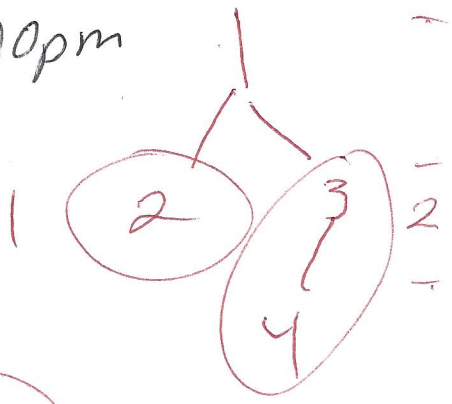
```
Merge (H1, H2) {
  if H1 is empty, return H2
  else if H2 is empty, return H1
  else let newroot = min(root(H1), root(H2))
    if newroot == root(H1)
      let ST1 = H1.left
          ST2 = H1.right
          ST3 = H2
    else
      let ST1 = H2.left
          ST2 = H2.right
          ST3 = H1
    if ST1.height >= ST2.height && ST1.height >= ST3.height
      new Heap (newroot, ST1, Merge (ST2, ST3))
    else if ST2.height >= ST1.height && ST2.height >= ST3.height
      new Heap (newroot, ST2, Merge (ST1, ST3))
    else // ST3 is largest
      new Heap (newroot, ST3, Merge (ST1, ST2))
}
```

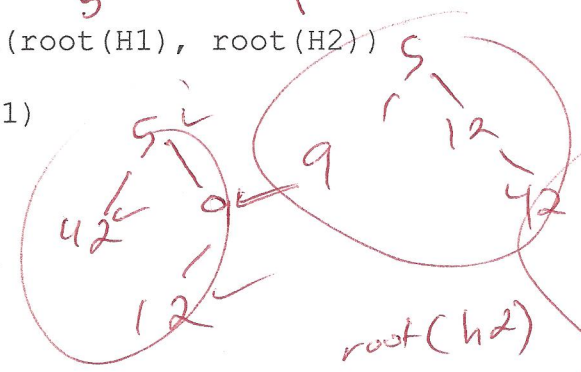


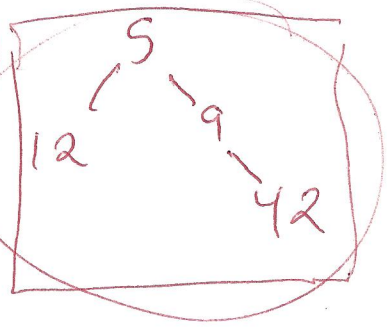
```
Merge (H1, H2) {

   if H1 is empty, return H2

   else if H2 is empty, return H1

   else let newroot = min(root(H1), root(H2))

     if newroot == root(H1)

       let ST1 = H1.left
           ST2 = H1.right
           ST3 = H2

     else

       let ST1 = H2.left
           ST2 = H2.right
           ST3 = H1

     if ST1.height >= ST2.height && ST1.height >= ST3.height

       new Heap (newroot, ST1, Merge (ST2, ST3))

     else if ST2.height >= ST1.height && ST2.height >= ST3.height

       new Heap (newroot, ST2, Merge (ST1, ST3))

     else // ST3 is largest

       new Heap (newroot, ST3, Merge (ST1, ST2))

}
```

1:00pm

terminating







root(h2)

newroot == ~~tmp~~

```
#BinTree //elsewhere
free=heap1.addElt(3);
addEltTester(heap1, 3, free)
```