Marinel Tinnirello
10/11/2019
CMPT 435 - Assignment #6

1)
   a) Run a for (i = 0; start <= end; i++)
      If x = A[i], increment the count
      If x < A[i], change start to mid + 1
      Else if x > A[i], change end to end - 1
      If x = -1, the x isn't in the array
   b)
```
for ( i = 0; start <= end; i++ )                    // n
    if ( x == A[i] ) { Output: count++ }            // 1
    if ( x < A[i] ) { Output: start = mid + 1 }     // 1
    else { Output:  end = mid - 1 }                 // 1
    if ( x == -1 ) { Output: return -1 }            // 1
```
   c) O(n) = log(n)

2)
   a) Take the stack of coins and split the amount in equal half (if even), split into 3 equal groups (if odd).
      Whichever side is heavier, discard the lighter side and split the heavier side into equal halves.
      Continue this process until you only have 2 coins left.
      Whichever the heavier one is the outlier coin.
   b) O(n) = n*log(n)

3)
   a) O(n)
   b) Using Binary Search on a sorted Linked List would give us the same time complexity as using a Linear Search. This because even if the Linked List is sorted, it isn't necessarily contiguous in memory, so it will take roughly the same time to perform a Binary Search as it would Linear.

4)
   a) Check if x is either 0 or 1. If so, return x.
      Run a while (start <= end)
      If mid^2 is equal to x, the number is a perfect square
      If mid^2 is less than x, change the start to mid + 1 and the answer to mid
      Else if mid^2 is greater than x, change the end to mid - 1
   b)
```
if ( x == 0 || x == 1 ) { Output: return x }        // 1
while ( start <= end )                              // n
    mid = (start + end) / 2                         // 1
    if ( mid * mid == x ) { Output: return mid }    // 1
    if ( mid * mid > x )                            // n
        start = mid + 1                             // 1
        ans = mid                                   // 1
```

```
        else                                        // n
                end = mid + 1                       // 1
```

c) O(n) = log n