Names: ______________________________ | Worksheet #6

CS Logins: ____________________________ | Binary Search Activities

As always, sit with a partner and work through these together.

Activity #1. Brainstorm different approaches to checking whether or not a particular name is in a phonebook, keeping in mind the runtime of each solution (reading any word on any page and comparing any 2 words to determine which comes first alphabetically are both constant time operations- O(1))

Activity #2. You and a partner will hand simulate a simple example of the in-place binary search algorithm.

```
function binarySearch(A, lo, hi, x):
// Input: A – a sorted array
//        lo, hi – two valid indices of the array
//        x – the item to find
// Output: true if x is in the array between lo and hi, inclusive,
//         else false
if lo >= hi:
       return A[lo] == x
mid = (lo + hi) / 2
if A[mid] == x:
       return true
if A[mid] < x:
       return binarySearch(A, mid + 1, hi, x)
if A[mid] > x:
       return binarySearch(A, lo, mid – 1, x)
```

Please copy the array and x value that Seny has written on the board.

A = [, , , , , ,]
x =

For each recursive call, fill in the chart to indicate what lo, hi, mid, and A[mid] are. Then compare these values (following the in-place algorithm pseudocode line by line) to determine whether the return statement would be the result of another recursive call (in which case you would circle recurse and continue filling out the next row) or if it would be true/false (in which case you would circle the return value). *Note: There may be many more rows than necessary.*

Recursive Call	LO	HI	MID	A[MID]	RETURN?
1	0	6			RECURSE/TRUE/FALSE
2					RECURSE/TRUE/FALSE
3					RECURSE/TRUE/FALSE
4					RECURSE/TRUE/FALSE
5					RECURSE/TRUE/FALSE
6					RECURSE/TRUE/FALSE
7					RECURSE/TRUE/FALSE

Names: ______________________________ Worksheet #6

CS Logins: ____________________________

Binary Search Activities

As always, sit with a partner and work through these together.

Activity #1. Brainstorm different approaches to checking whether or not a particular name is in a phonebook, keeping in mind the runtime of each solution (reading any word on any page and comparing any 2 words to determine which comes first alphabetically are both constant time operations- O(1))

Activity #2. You and a partner will hand simulate a simple example of the in-place binary search algorithm.

```
function binarySearch(A, lo, hi, x):
// Input: A – a sorted array
//        lo, hi – two valid indices of the array
//        x – the item to find
// Output: true if x is in the array between lo and hi, inclusive,
//         else false
if lo >= hi:
       return A[lo] == x
mid = (lo + hi) / 2
if A[mid] == x:
       return true
if A[mid] < x:
       return binarySearch(A, mid + 1, hi, x)
if A[mid] > x:
       return binarySearch(A, lo, mid – 1, x)
```

Please copy the array and x value that Seny has written on the board.

A = [, , , , , ,]

x =

For each recursive call, fill in the chart to indicate what lo, hi, mid, and A[mid] are. Then compare these values (following the in-place algorithm pseudocode line by line) to determine whether the return statement would be the result of another recursive call (in which case you would circle recurse and continue filling out the next row) or if it would be true/false (in which case you would circle the return value). *Note: There may be many more rows than necessary.*

Recursive Call	LO	HI	MID	A[MID]	RETURN?
1	0	6			RECURSE/TRUE/FALSE
2					RECURSE/TRUE/FALSE
3					RECURSE/TRUE/FALSE
4					RECURSE/TRUE/FALSE
5					RECURSE/TRUE/FALSE
6					RECURSE/TRUE/FALSE
7					RECURSE/TRUE/FALSE