DS & A Final

For the following five problems, write your answers out in 1-2 paragraphs (no coding required):

1. How are hash tables used to implement JavaScript objects?

2. Why is knowing Big O runtime important? Give an example of

3. What is the difference between a contiguous array and a non-contiguous array?

4. You are building a phone book app that has a large amount of phone numbers. You want your users to be able to look up numbers very quickly, even if you have millions of entries. What data structure(s) would be helpful and why?

5. How would you go about implementing a queue in JavaScript? What methods would need to include? Do not write code. Just explain in words the strategy you would use for creating one.

For the following three problems, identify the Big O runtime of each:

6. Big O Runtime 1:

```
function problem1 (arr) {
  var sum = 0;
  for(var i = 0; i < arr.length; i++) {
    for(var i = 0; i < arr.length; i++) {
      console.log(i);
    }
  }
  for(var i = 0; i < 10; i++) {
    sum += 1;
```

```
  }
  return sum;
}

problem1([1, 2, 3, 4, 5]);
```

7. Big O Runtime 2:

```
function problem2 (arr) {
  var sum = 0;
  for(var i = 0; i < arr.length; i++) {
    sum += i;
  }
  for(var i = 0; i < arr.length; i++) {
    sum += i;
  }
  return sum;
}

problem2([1, 2, 3, 4, 5]);
```

8. Big O Runtime 3:

```
function problem3 (arr) {
  for(var i = 0; i < 10; i++) {
    for(var j = 0; j < 10; j++) {
        console.log(k);
    }
  }
}

problem3([1, 2, 3, 4, 5]);
```

For the following problems, your responses will be evaluated on both their logic and their code. You may write your answers just in JavaScript,

or in both JavaScript and pseudocode. If you choose to write pseudocode, we will evaluate the logic of the pseudocode as part of your response. You can receive partial credit for a logically sound pseudocode response. To receive full credit, you must additionally include real code:

9. Given an array, write a function called 'removeDupes' that returns a new array of just the unique values. For full credit, your solution should be O(n)

```
removeDupes([1, 2, 3, 4, 5, 1, 2, 3]);
// [4, 5]
removeDupes(['cat', 'cat', 'dog', 4, 5, 5, 'bear']);
// ['dog', 4, 'bear']
```

10. Given an array, write a function called 'countTheLetters' that takes in a string and returns a count of each of the letters in the string. For full credit, your solution should be O(n)

```
countTheLetters('dog');
// {d: 1, o: 1, g: 1}
countTheLetters('elephant');
// {e: 2, l: 1, p: 1, h: 1, a: 1, n: 1, t: 1}
countTheLetters('llama');
// {l: 2, a: 2, m: 1}
```

11. Given an array, write a function called 'threeShortestWords' that returns a new array containing the three shortest words from the original array. For full credit, your solution should be O(n)

```
threeShortestWords(['one', 'two', 'three', 'four', 'six']);
// ['one', 'two', 'six']
threeShortestWords(['cat', 'dog', 'bear', 'rabbit', 'bat']);
// ['cat', 'dog', 'bat']
```

12. Write a function called 'searchLinkedList' that receives the head of a linked list and search value, and returns 'true' if that value is in the list, or 'false' if the value is not in the list

```
//if the linkedList contains a node with the value 'cat':
searchLinkedList(linkedListHead, 'cat'); //true
//if the linkedList does NOT contain 'cat':
searchLinkedList(linkedListHead, 'cat'); //false
```

13. Write a recursive function called 'recursiveZip' to zip two arrays (a 'zip' is when you take two arrays and return a single array of arrays, where each inner array contains a pair of values - one value from the first array, and another value from the second). If one array is longer than the other, you should return 'null' as part of your pair. You should not use 'for' loops, 'while' loops, 'map', 'filter', 'reduce', or any other non-recursive array methods.

```
var arr1 = [1, 2, 3], [4, 5, 6];
var arr2 = ['cat', 'dog'], ['bear', 'frog', 'rabbit'];
recursiveZip(arr1);
//[[1, 4], [2, 5], [3, 6]]
recursiveZip(arr2);
//[['cat', 'bear'], ['dog', 'frog'], [null, 'rabbit']]
```