CSCE 221 Cover Page
Homework Assignment #2
Due July 25 at 23:59 pm to eCampus

First Name: Jonathan Last Name: Westerfield UIN: 224005649

User Name: jgwesterfield E-mail address: jgwesterfield@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources	Peer Teachers		
People	Lauren Kleckner		
Web pages (provide URL)			
Printed material			
Other Sources			

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Your Name: Jonathan Westerfield Date: 7/19/17

Homework 2

due July 25 at 11:59 pm.

1. (15 points) Write a recursive function that counts the number of nodes in a singly linked list. Write a recurrence relation that represents your algorithm. Solve the relation and obtain the running time of the algorithm in Big-O.

```
int countNodes(node *currentNode) {
    if(currentNode == NULL)
       return(0);
    return(1 + countNodes(currentNode->next));
    }
```

$$a_n = a_{n-1} + 1$$

$$O(n)$$

2. (15 points) Write a recursive function that finds the maximum value in an array of int values without using any loops. Write a recurrence relation that represents your algorithm. Solve the relation and obtain the running time of the algorithm in Big-O.

```
int maxNum(int x[], int lowIndex, int highIndex) {
    if (lowIndex == highIndex) {
       return x[lowIndex];
    }
    if(lowIndex < highIndex) {
       if(x[lowIndex] < x[highIndex]) {
         return maxNum(x, lowIndex + 1, highIndex);
       }
       return maxNum(x, lowIndex, highIndex - 1);
    }
}
```

$$F(n) = F(n-1) + 1$$

$$F(n-1) = F(n-2) + 1$$

$$F(n-2) = F(n-3) + 1$$

$$F(n-3) = F(n-4) + 1$$

$$F(k) = F(n-(k-1)) + kn$$

$$F(1) = 1$$

$$O(n)$$

3. (10 points) What data structure is most suitable to determine if a string s is a palindrome, that is, it is equal to its reverse. For example, "racecar" and "gohan gasalamiimalasagnahog" are palindromes. Justify your answer. Use Big-O notation to represent the efficiency of your algorithm.

   The stack data structure would be the best data structure to use for determining if a string is a palindrome. To do this, you would push the characters string onto the stack starting from the beginning of the string. From there, you would pop off elements and compare those elements (which would be starting from the end of the string) to the elements at the beginning of the string. This require going through the string 2 times. One time to push on the stack and the other time to pop and compare. This means that this stack implementation would be $O(n)$.

1. (10 points) Write a pseudocode to implement the stack ADT using two queues. What is the running time of the push and pop functions in this case?

```
Queue1
Queue2
Push(x) // puts element on the stack (which is queue 1)
    Queue1.enqueue(x);
Pop() // takes element out of queue 1 and puts it into queue 2
    if(¬Queue1.isempty())
        Queue2.enqueue(Queue1.dequeue)
    else
        throw exception
```

The Push function is $O(1)$ because there is only one operation done. The pop function is also $O(1)$ since there is only 1 function that can happen.

1. (10 points) What is the best, worst and average running time of quick sort algorithm? Provide arrangement of the input and the selection of the pivot point at every case. Provide a recursive relation and solution for each case.

   Best runtime: $\Omega(nlogn)$. Average case: $\theta(nlogn)$. Worst case: $O(n^2)$.

   Best case is when the list is sorted and the pivot is in the middle of the list. The recursive relation for the best case scenario is:

$$T(n) = 2T(\frac{n}{2}) + n$$

$$T(n) = 2(2(\frac{n}{4}) + \frac{n}{2}) + n$$

$$T(n) = 2(2(2(\frac{n}{8}) + \frac{n}{4} + \frac{n}{2}) + n$$

$$\therefore$$

$$= 2^k T(\frac{n}{2^k}) + i * n$$

$$\therefore$$

$$= n * 0 + log_2 n * n$$

$$\therefore O(nlogn)$$

   The average case for quick sort is anytime that isn't the best case or the worst case and the recurrence relation is the same as the best case recurrence relation. The worst case scenario is when the list is either unsorted and the pivot point is at either end of the list or the list is sorted in reverse order and the pivot point is at the ends of the list. The reccurence relation for the worst case scenario is:

$$T(n) = T(n-1) + n$$

$$T(n) = T(n-2) + (n-1) + n$$

$$T(n) = T(n-3) + (n-2) + (n-1) + n$$

$$1 + 2 + 3 + ... + (n-3) + (n-2) + (n-1) + n \therefore Arithmetic$$

$$\therefore$$

$$T(n) = \frac{n(n+1)}{2}$$

$$\therefore O(n^2)$$

2. (10 points) What is the best, worst and average running time of merge sort algorithm? Use two methods for solving a recurrence relation for the average case to justify your answer.

Best runtime: $\Omega(nlogn)$. Average case: $\theta(nlogn)$. Worst case: $O(nlogn)$.

Recurrence relation of Merge sort: $T(n) = 2T(\frac{n}{2}) + n$

Solving the relation using Master Theorem: $a = 2$, $b = 2$, $c = 1$, $d = 1$

By using masters theorem, we determine that since $a = b^d$ since $2 = 2^1$. Therefore the function has a run time of $O(nlogn)$.

Also we can use the substitution method to solve the recurrence relation.

$$T(n) = 2T(\frac{n}{2}) + n$$

$$T(n) = 2(2T(\frac{n}{4})) + 2n$$

$$T(n) = 2(2(T(\frac{n}{8}))) + 3n$$

$$T(n) = 2^k T(\frac{n}{2^k}) + kn$$

$$k = log_2 n$$

$$\therefore$$

$$T(n) = 2^{log_2 n} T(1) + log_2(n)n = n * 0 + nlogn$$

$$\therefore O(nlogn)$$

3. (10 points) R-10.17 p. 493 For the following statements about red-black trees, provide a justification for each true statement and a counterexample for each false one.

   (a) A subtree of a red-black tree is itself a red-black tree.

   This is false. While a red black tree is a self balancing binary tree, the colors of the nodes don't always alternate from level to level. The condition for the colors of the nodes is that there are always the same number of black nodes from the root to any nodes on the same level. This means that it is possibly that the colors of one side of the tree can be different than the other side of the tree. Also, any subtree rooted at a red node cannot be a red-black tree because this contradicts what makes a red black tree.

   (b) The sibling of an external node is either external or it is red.

   This is true. If an external node had a black internal sibling, then these 2 sibligns woiuld not have the same black depth, which contradicts red-black trees having all leaves be the same black depth.

   (c) There is a unique (2,4) tree associated with a given red-black tree.

   This is true. When merging each black node with its red children, the combinations will either form a 2, 3, or 4 node. They fall into set configurations because the tree is being "condensed" in a way.

   (d) There is a unique red-black tree associated with a given (2,4) tree.

   This is false. A (2,4) tree can have multiple red-black tree configurations.

4. (10 points) R-10.19 p. 493 Consider a tree T storing 100,000 entries. What is the worst-case height of T in the following cases?

   (a) T is an AVL tree.

$$h(T) = O(1.44logn) = log_2(100,000) \approx 23$$

   (a) T is a (2,4) tree.

$$h(T) = O(log_2 n) = log_2(100,000) \approx 16$$

(a) T is a red-black tree.

$$h(T) = O(2log_2 n) = log_2(100,000) \approx 33$$

(b) T is a binary search tree.

$$h(T) = O(n) = 100,000$$

5. (10 points) R-9.16 p. 418 Draw an example skip list that results from performing the following series of operations on the skip list shown in Figure 9.12: `erase(38)`, `insert(48,x)`, `insert(24,y)`, `erase(55)`. Record your coin flips, as well.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	original skip list					s5	-∞												∞		
2						s4	-∞		15										∞		
3						s3	-∞		15		27				42			55	∞		
4						s2	-∞		15		27	31			42			55	∞		
5						s1	-∞	12	15	✓	27	31	38		42	44		55	∞		
6						s0	-∞	12	15	20	27	31	38	39	42	44		55	∞		
7																					
8	after erase(38)					s5	-∞												∞		
9						s4	-∞		15										∞		
10						s3	-∞		15		27			42				55	∞		
11						s2	-∞		15		27	31		42				55	∞		
12						s1	-∞	12	15		27	31		42	44			55	∞		
13						s0	-∞	12	15	20	27	31	39	42	44	50		55	∞		
14																					
15	after insert(48,x)					s5	-∞													∞	
16	x =4 coin tosses					s4	-∞		15							48				∞	
17						s3	-∞		15		27			42		48			55	∞	
18						s2	-∞		15		27	31		42		48			55	∞	
19						s1	-∞	12	15		27	31		42	44	48			55	∞	
20						s0	-∞	12	15	20	27	31	39	42	44	48		50	55	∞	
21																					
22	after insert(24,y)					s5	-∞														∞
23	y = 3 coin tosses					s4	-∞		15									48			∞
24						s3	-∞		15		24	27			42			48		55	∞
25						s2	-∞		15		24	27	31		42			48		55	∞
26						s1	-∞	12	15		24	27	31		42	44		48		55	∞
27						s0	-∞	12	15	20	24	27	31	39	42	44		48	50	55	∞
28																					
29	after erase(55)					s5	-∞													∞	
30						s4	-∞		15									48		∞	
31						s3	-∞		15		24	27			42			48		∞	
32						s2	-∞		15		24	27	31		42			48		∞	
33						s1	-∞	12	15		24	27	31		42	44		48		∞	
34						s0	-∞	12	15	20	24	27	31	39	42	44		48	50	∞	