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

First Name Hunter Last Name Cleary UIN 625001547

User Name hncleary E-mail address hncleary@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/http://aggiehonor.tamu.edu/

Type of sources			
People			
Web pages (provide URL)	Links Below		
Printed material	Data Structures and Algorithms (Textbook)		
Other Sources			

https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms)
https://www.geeksforgeeks.org/quick-sort/
https://www.geeksforgeeks.org/merge-sort/
https://en.wikipedia.org/wiki/Skip_list
http://www.cplusplus.com/reference/queue/queue/
https://www.cs.auckland.ac.nz/software/AlgAnim/AVL.html
https://en.wikipedia.org/wiki/Recurrence_relation

I certify that I have listed all the sources that I used to develop the solutions/codes 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 Hunter Cleary Date 3-22-18

Homework 2

due March 25 at 11:59 pm to eCampus.

1. (15 points) Describe how to implement the stack ADT using two queues.

Write a C++ function that implements your solution. You can use the C++ STL queue container.

Listing 1: Stack ADT Implementation

```
#include <isostream>
#include <queue>

void push(T item,queue<T> queue) {
   queue.push_back(item);
}
T pop(queue<T> queue) {
   queue queue2;
   while (queue1.size() > 1) { // copy all items except for one
      queue2.push_back(queue.pop_front());
   }
   T item = queue.pop_front(); //pop lat item in first queue
   while (queue2.isEmpty()) { // copy
      queue.push_back(queue2.pop_front());
   }
   return item;
}
isEmpty(queue<T> queue) {
   if (queue.empty() == true) { // check to se if queue is empty using empty()
      return true;
   }
   else{
      return false;
   }
}
```

What is the running time of the push and pop functions in this case?
The running time of push is O(1); it only performs one operation. The running time of pop is O(n); it must cycle through and copy the items present in the first queue (except for one) and then cycle through and copy the items from a secondary queue into the original. $f(x) = 4n = O(n)$

2. (15 points) Write a recursive function in C++ that counts the number of nodes in a singly linked list.

   (a) Test your function using different singly linked lists. Include the code and screenshots with testing cases.

Listing 2: Stack ADT Implementation

```
int count(*ListNode node) {
   if (node == NULL) {
      return 0;
   }
   else if (node != NULL) {
      return 1 + count(node->next);
   }
}
```

   (b) Write a recurrence relation that represents your algorithm.

$T(n) = T(n-1) + 1$
$T(0) = 1$

(c) Solve the relation using the iterating or a recursive tree method to obtain the running time of the algorithm in Big-O notation.

$T(n) = T(n-1) + 1$
$= T(n-2) + 2$
$= T(n-3) + 3$
$= T(n-4) + 4$
$= T(n-5) + 5$
$\equiv$T(n-k) + k
The maximum of k is n
$\equiv$T(0) + n
$= 1 + n$
T(n) is O(n)

3. (15 points) Write a C++ recursive function that finds the maximum value in an array of integers without using any loops.

   (a) Test your function using different input arrays. Include the code and screenshots with testing cases.

Listing 3: Stack ADT Implementation

```
arrayMax(int list[], int size, int max) {
   int temp_max = max;
   if (size == 0) {
      return temp_max;
   }
   if (list[size - 1] > temp_max) {
      temp_max = A[size - 1];
   }
   arrayMax(list, size - 1, temp_max);
}
```

(b) Write a recurrence relation that represents your algorithm. Solve the relation and obtain the running time of the algorithm in Big-O notation.

$T(n) = T(n-1) + 1$
$T(0) = 1$

$T(n) = T(n-1) + 1$
$= T(n-2) + 2$
$= T(n-3) + 3 \cdots$
≡T(n-k) + k
The maximum of k is n
≡T(0) + n
= 1 + n
T(n) is O(n)

4. (15 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.

$Best = O(nlogn), Average = O(nlogn), Worst = O(n^2)$

Best - The best case for the algorithm occurs when the pivot point is located in the middle. This means that the size of both sides are equal. Each recursive call will process a list that is half the size.

$T(1) = 0$
$T(n) = 2T(n/2) + n$
$T(n/2) = 2T(n/4) + n/2$
$T(n/4) = 2T(n/8) + n/4$
$T(n/8) = 2T(n/16) + n/8$
$T(n)$ is $O(n \log n)$

Average - Each pivot point has is selected between %25 and %75, leading to an average case of %50. Since the average point is the middle, the average case is equal to the best case.

Same as best case.

Worst - The worst case will occur when the pivot element is the smallest or largest in the list. If this happens during every recursive step, the function will take $O(n^2)$ time.

$T(1) = 0$
$T(n) = T(n-1) + n$
$T(n-1) = T(n-2) + n - 1$
$T(n-2) = T(n-3) + n - 2$
$T(n-3) = T(n-4) + n - 3$
$= (n(n-1))/2$
$T(n)$ is $O(n^2)$

5. (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.

$Best = O(nlogn), Average = O(nlogn), Worst = O(nlogn)$

Iterative Method
$T(1) = 0$
$T(n) = 2T(n/2) + n$
$T(n/2) = 2T(n/4) + n/2$
$T(n/4) = 2T(n/8) + n/4$
$T(n/8) = 2T(n/16) + n/8$
$T(n)$ is $O(n \log n)$

Master Theorem
$T(n) = 2T(n/2) + n$
$f(n) = aT(n/b) + f(n)$, $c = \log_b a, a = 2, b = 2, k = 0$
is $O(n^2)$

6. (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.

   False - The root of a Red-Black tree is black. The subtree of a Red-Black tree can start with either a red or a black root node. Therefore it is not always true.

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

   True - Black depth must be the same for the external node, so the statement is true.

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

   True - Each node in a given Red-Black tree is represented uniquely in a (2,4) tree.

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

False - A (2,4) tree can be represented 2 different ways in a Red-Black tree.

7. (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.

$ceiling(2)(\log_2 100000 + 1) = 34$

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

$ceiling(\log_2 100000 + 1) = 17$

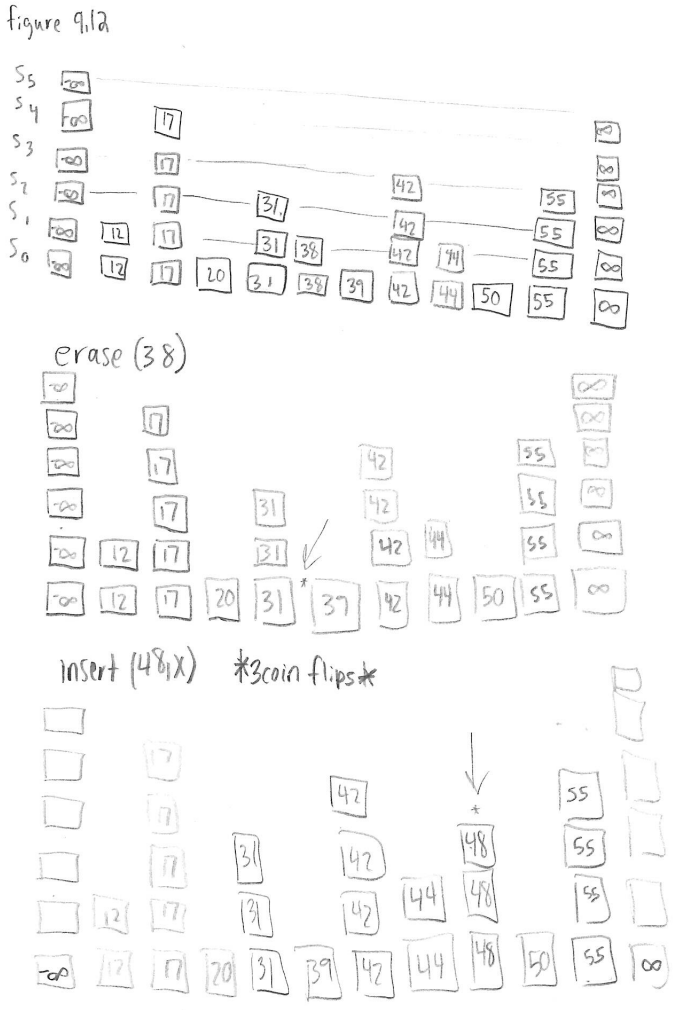
(c) T is a red-black tree.

$ceiling((2)\log_2 100000 + 1) = 34$

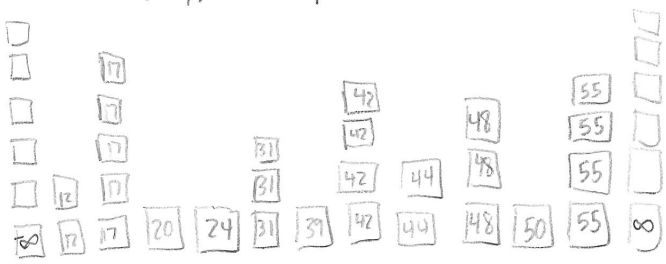
(d) T is a binary search tree.

n
$100,000$

8. (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.



insert (24, y) *1 flip*



erase (55)