CSCE 221 Cover Page
Homework #2
Due July 24 at 23:59 pm to eCampus

First Name McKenzie Last Name Burch UIN 225005240

User Name mburch13gig-em E-mail address mburch13gig-em@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			
People	Sara Wild	Caio Monteiro (TA)	
Web pages (provide URL)	https://www.wolframalpha.com/ https://stackoverflow.com/		
Printed material	Textbook		
Other Sources	Personal Notes		

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 McKenzie Burch Date July 23, 2018

Homework 2

due July 24 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.

```
countNodes(head){
    if(head == NULL)
        return 0;
    else
        return 1 + countNodes(head->next);
}
```

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

$T(n) = 0$

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

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

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

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

General Form: $T(n) = T(n-k) + 1$ where $k = n$

Therefore $T(n) = n$ which has a runtime of $O(n)$

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.

```
maxValue(array, int n){
    n = array.size();
    if(n == 1)
        return n;
    else
        return max(n-1, maxValue(array,n-1));
}
```

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

$T(n) = 1$

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

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

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

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

General Form: $T(n) = T(n-k) + 1$ where $k = n$

Therefore $T(n) = 1 + n$ which has a runtime of $O(n)$

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 data structure most suitable to determine if a string is a palindrome is a stack, because when extracting data from a stack the string output is reversed. Therefore if the *input* == *output* then the string is a palindrome. The efficiency of the algorithm is $O(n)$ because with n characters there will be n comparisons between the strings.

4. (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?

   - The running time of the push function would be $O(1)$ and would be the same code because both stack and queue insert at the end of list.
   - The running time of the pop function would be $O(n)$

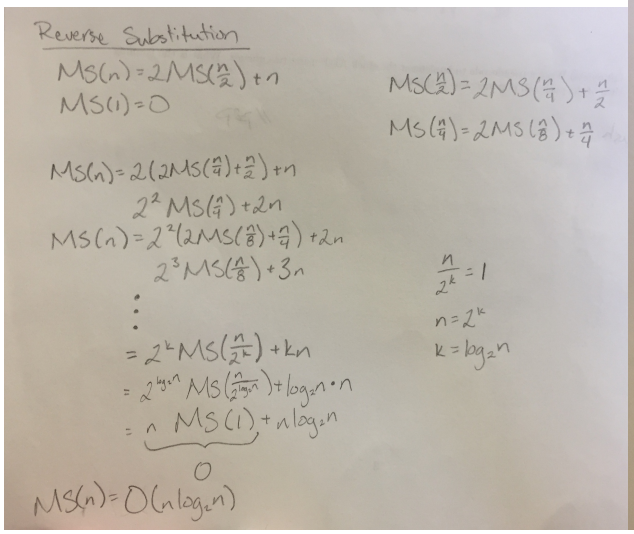
```
//pop using 2 queues
Queue q1, q2;
int temp;
while q1 is not empty:
   for i = 0 → n-2:
     q1.dequeue() = q2.enqueue(i);
   temp = q1.dequeue();
   q1 = q2;
   return temp;
```

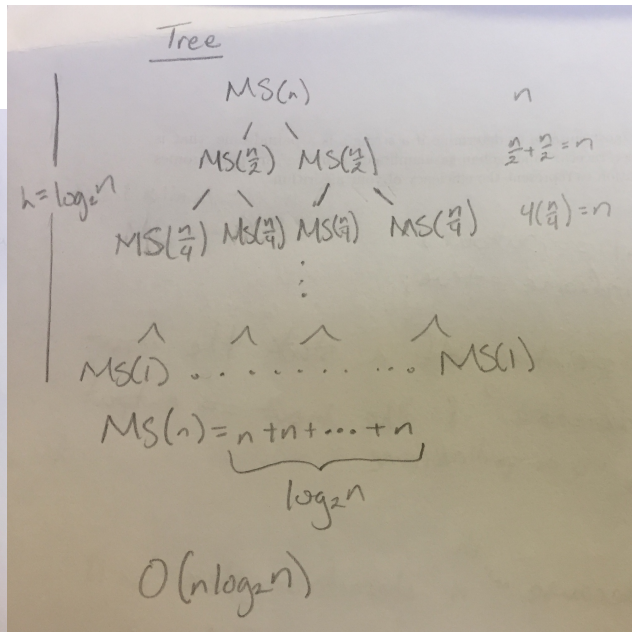
5. (10 points) What is the best, worst and average running time of quick sort algorithm? Provide arrangement of the input and the selection of the pivot point at every case. Provide a recursive relation and solution for each case.

   - Best Case: when partitions are evenly divided and the pivot point is the median of each partition
     - Recurrence Relation: $QS(n) = 2QS(n/2) + n$
     - Base Case: $QS(1) = 0$
     - Running Time: $O(nlog_2n)$
   - Worst Case: always choosing a pivot point that is the smallest or largest number making the partitions $n-1$ each recursion
     - Recurrence Relation: $QS(n) = QS(n-1) + QS(1) + n = QS(n-1) + n$
     - Base Case: $QS(1) = 0$
     - Running Time: $O(n^2)$
   - Average Case: randomly choosing a pivot point
     - Recurrence Relation: $QS(n) = QS(\alpha n) + QS((\alpha - 1)n) + n$
     - Base Case: $QS(1) = 0$
     - Running Time: $O(nlog_2n)$

6. (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, Worst, and Average Case:
     - Recurrence Relation: $MS(n) = 2MS(n/2) + n$
     - Base Case: $MS(1) = 0$
     - Running Time: $O(nlog_2n)$





7. (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.

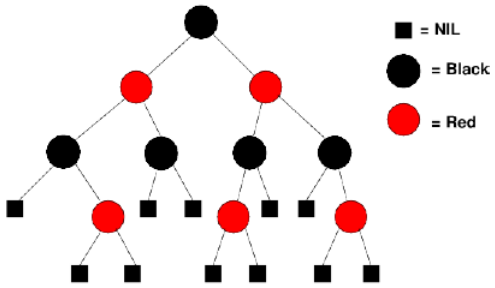
   (a) A subtree of a red-black tree is itself a red-black tree.

   - False, because each root node must be black and the children of a black node must be either red or black. Therefore, if you take the left or right subtree from the children of the root node, then the root would be red and therefore not a red-black tree.

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

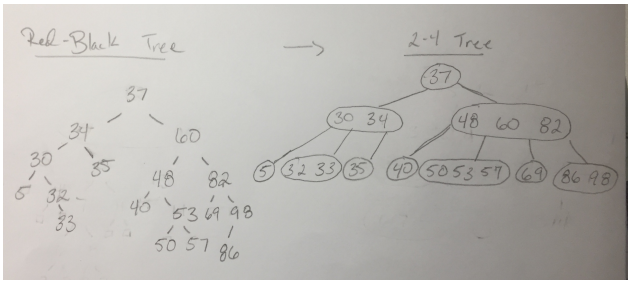
   - True, because the children of a black node is either an external node or a red node.

   Example from lecture slides:



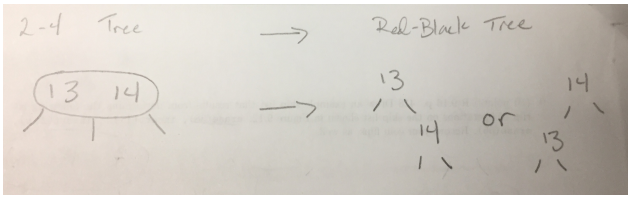
   (c) There is a unique (2,4) tree associated with a given red-black tree.

   - True, because you can make a 2-4 tree with any red-black tree.



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

   - False, because with a 2-4 node can have two options of creating a red-black tree.



8. (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.

      i. Height:
      $h \leq 1.44log_2(n+1) - 1.328$
      $n = 100,000$
      $h \leq 1.44log_2(100000+1) - 1.328$
      $h \leq 22.59$

      ii. Worst Case Height:
      $O(log_2 n)$

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

      i. Height:
      $h \leq log_2(n+1)$
      $n = 100000$
      $h \leq log_2(100000+1)$
      $h \leq 16.61$

      ii. Worst Case Height:
      $O(log_2 n)$

   (c) T is a red-black tree.

      i. Height:
      $h \leq 2log_2(n+1)$
      $n = 100000$
      $h \leq 2log_2(n+1)$
      $h \leq 33.22$

      ii. Worst Case Height:
      $O(log_2 n)$

   (d) T is a binary search tree.

      i. Height:
      $n = 100000$

      ii. Worst Case Height:
      $O(n)$

9. (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.

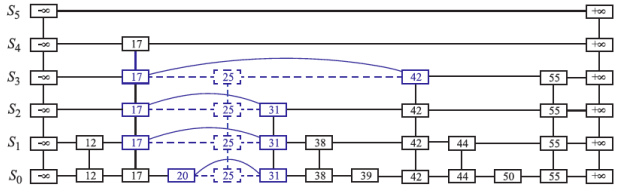
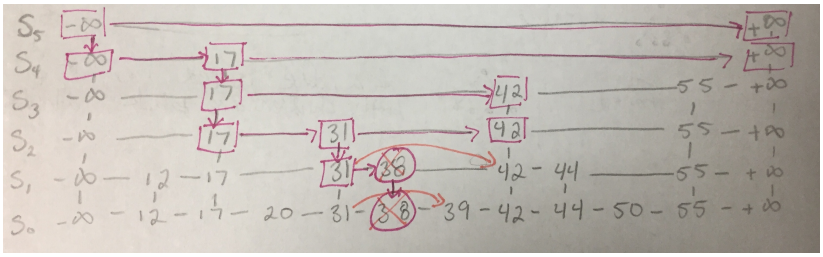
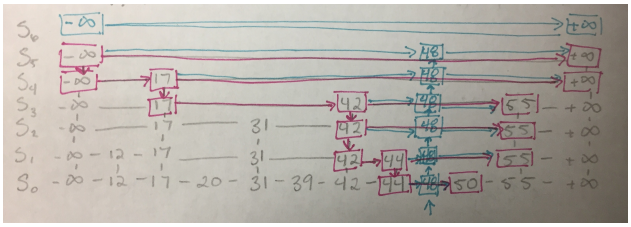


Figure 9.12: Removal of the entry with key 25 from the skip list of Figure 9.11. The positions visited after the search for the position of S_0 holding the entry are highlighted in blue. The positions removed are drawn with dashed lines.

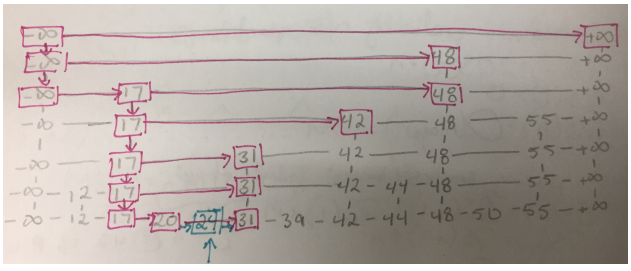
```
erase(38)
    Search(38)
        if not 38
            return null
            get p of 38
        remove 38
```



```
    insert(48,x)
    coinFlip:  6
    Heads:     5
    Tails:     1
```



```
    insert(24,y)
    coinFlip:  1
    Heads:     0
    Tails:     1
```



```
erase(55)
    Search(55)
        if not 55
            return null
            get p of 55
        remove 55
```