CSCE 221 Cover Page

Homework Assignment #2

First Name Alexander **Last Name** Kaiser **UIN** 924007333

User Name ALKYAYZZZ **E-mail address** alkyayzzz@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 on Aggie Honor System Office website: `http://aggiehonor.tamu.edu/`

Type of sources				
People				
Web pages (provide URL)				
Printed material	Data Structures and Algorithms in C++ M.T. Goodrich, R. Tamassia and D. Mount			
Other Sources				

I certify that I have listed all the sources that I used to develop the solutions/codes in the submitted work.
On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.

Your Name Alexander Kaiser Date 10/11/16

Homework 2

due October 27 at 11:59 pm
Submission to e-Campus

1. (10 points) Describe (in pseudo code) how to implement the stack ADT using two queues. What is the running time of the push and pop functions in this implementation?
   The stack can be implemented in two different ways.

   -push://$O(1)$
   –enqueue in queue 1.
   -pop:$O(n)$
   –while size of queue 1 is greater than 1, pipe dequeued items from queue 1 into queue 2.
   –dequeue and return the last item from queue 1, then switch the names of queue 1 and queue 2.

   -push://$O(n)$
   –enqueue in queue 2
   –enqueue all the items from queue 1 to queue 2, then switch the names of the two queues.
   -pop://$O(1)$
   –dequeue queue 1.

2. (10 points) Solve C-5.8 on p. 224
   Algorithm evaluatePostfixExpression(L)
   Input: A postfix expression stored in a list L
   Output: The result of the postfix expression
   {
   create an empty stack S
   while (L.hasNext())
   if(x is a number)
   S.push(x);
   else {
   a = S.pop();
   b = S.pop();
   S.push(b x a);
   }
   }
   return S.pop();
   }

3. (10 points) Linked list questions.

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

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

   (b) Write a recurrence relation that represents the running time for your algorithm.

   T(n) = C1 when n = 1

   T(n) = T(n - 1) + C2 when n > 1

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

   (c) Solve this relation and provide the classification of the algorithm using the Big-O asymptotic notation.

   T(n) = T(n - 1) + 1 = T(n - 2) + 2 = ... = 1 + n = O(n)

4. (10 points) Write a recursive function that finds the maximum value in an array of integers without using any loops.

```
#include <algorithm>
int maxnum(int i[], int index)
{
if(index > 0)
return max(i[index], maxnum(i, index - 1));
else return i[0];
}
```

   (a) Write a recurrence relation that represents running time of your algorithm.

   T(1) = 0

   T(n) = T(n-1) + 1 + T(1) when n > 1

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

   (b) Solve this relation and classify the algorithm using the Big-O asymptotic notation.

   T(n) = T(n - 1) + 1 = T(n - 2) + 2 = ... = 1 + n = O(n)

5. (10 points) Consider the quick sort algorithm.

   (a) Provide an example of the inputs and the values of the pivot point for the best, worst and average cases for the quick sort.
   -Worst case has a pivot as the first or last element of the array with the array already sorted.
   Ex: 5(Pivot),10,15,20,25,30,35
   -Best case has a pivot as the middle element of the array with the array already sorted.
   Ex: 5,10,15,20(Pivot),25,30,35
   -Average case has a pivot as a randomly chosen element of the array with the array not sorted previously.
   Ex: 25,20,35,5,30,10,15(Pivot)

   (b) Write a recursive relation for running time function and its solution for each case.
   -Worst case:T(n) = T(n - 1) + c(n) + d(n)
   = T(n - 1) + O(n) + O(n)
   = T(n - 1) + O(n)
   = O(n^2)
   -Best case:T(n) = a * T(n/b) + c(n) + d(n)
   = 2 * T(n/2) + O(n) + O(n)
   = 2 * T(n/2) + O(n)
   = O($nlog_2n$)
   -Average case:T(n) = T(n * a) + T(n * (1 - a)) + O(n) $0 < a < 1$
   = O($nlog_2n$)
   Somewhere in between the best and worst case

6. (10 points) Consider the merge sort algorithm.

   (a) Write a recurrence relation for running time function for the merge sort.
   T(n) = a * T(n/b) + c(n) + d(n)
   = 2 * T(n/2) + O(n) + O(n)
   = 2 * T(n/2) + O(n)

(b) Use two methods to solve the recurrence relation.
Substitution: T(n) == c * $nlog_2n$
T(n) = 2 * (c * n/2 * $log_2(n/2)$) + n
= c * n * $log_2(n/2)$ + n
= c * n * (log_2n - 1) + n
= c * n * log_2n - c * n + n
$\leq$ c * $nlog_2n$
True when c $\geq$ 1

Iterative:T(n) = 2 * T(n/2) + n
= 2 * (2 * T(n/4) + n/2) + n
= 2 * (2 * (2 * T(n/8) + n/4) + n/2) + n
...
$= 2^i * T(n/2^i) + i * n)$
Max value of i == log_n
$= 2^{logn} * T(n/2^{logn})$ + nlogn
$2^{logn} == n$
$n/2logn == 1$
= n * T(1) + nlogn
= n * c + nlogn
= O(nlogn)

(c) What is the best, worst and average running time of the merge sort algorithm? Justify your answer.
O(nlogn) for the best, worst and average running time for the merge sort because the array is divided into the same number of subgroups regardless of how the array is already sorted.

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.
The statement is false, the root node of a subtree of a red-black tree is either a red or a black node. Thus, a subtree of a red-black tree may not be another red-black tree.

(b) The sibling of an external node is either external or it is red.
The statement is true, an external black node has an internal sibling and the black depth has a depth of h, the black depths of the external nodes below the black internal siblings is $h + 1$, this violates the depth property of a red-black tree. Thus, an external node must have the same number of black ancestors.

(c) There is a unique (2,4) tree associated with a given red-black tree.
The statement is true, if each black node is merged with its red children, a 2-3-4 node is formed depending upon the number of red children.

(d) There is a unique red-black tree associated with a given (2,4) tree.
The statement is false, each 3-node can be represented in one of two ways, it becomes a black node with a red child which becomes either a left or a right child.

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.
Let n(h) be the minimum number of internal nodes of an AVL tree with a height h.
n(h) = n(h - 1) + n(h - 2) + 1 (h $\geq$ 3)
n(h) = F_{h+2} - 1 where F_{h+2} is a fibonacci number.
$F_h = [a^h/5^{1/2}]$ where a = $(1 + 5^{1/2})/2$
Therefore, h = $\log(5^{1/2}(n(h)+1)/\log a$ - 2
h = 23 for an AVL tree storing 100000 entries.

(b) T is a (2,4) tree.
The height of a (2,4) tree storing n entries is at most log(n+1). Thus the worst case height of T is log(100000+1) = 16.

(c) T is a red-black tree.
The worst case height is 100000-1.

(d) T is a binary search tree.
The worst case height is 100000-1.

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.

erase(38)

-INF										INF
-INF		17								INF
-INF		17				42			55	INF
-INF		17		31		42			55	INF
-INF	12	17		31		42	44		55	INF
-INF	12	17	20	31	39	42	44	50	55	INF

insert(48,x)

-INF											INF
-INF								48			INF
-INF		17						48			INF
-INF		17				42		48		55	INF
-INF		17		31		42		48		55	INF
-INF	12	17		31		42	44	48		55	INF
-INF	12	17	20	31	39	42	44	48	50	55	INF

insert(24,y)

-INF												INF
-INF				24					48			INF
-INF		17		24					48			INF
-INF		17		24			42		48		55	INF
-INF		17		24	31		42		48		55	INF
-INF	12	17		24	31		42	44	48		55	INF
-INF	12	17	20	24	31	39	42	44	48	50	55	INF

erase(55)

-INF											INF
-INF				24					48		INF
-INF		17		24					48		INF
-INF		17		24			42		48		INF
-INF		17		24	31		42		48		INF
-INF	12	17		24	31		42	44	48		INF
-INF	12	17	20	24	31	39	42	44	48	50	INF

10. (10 points) R-9.7 p. 417

Draw the 11-entry hash table that results from using the has function, $h(k) = (3k + 5) \bmod 11$, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.

$(3(12)+5) \bmod 11 = 8$
$(3(44)+5) \bmod 11 = 5$
$(3(13)+5) \bmod 11 = 0$
$(3(88)+5) \bmod 11 = 5$
$(3(23)+5) \bmod 11 = 8$
$(3(94)+5) \bmod 11 = 1$
$(3(11)+5) \bmod 11 = 5$
$(3(39)+5) \bmod 11 = 1$
$(3(20)+5) \bmod 11 = 10$
$(3(16)+5) \bmod 11 = 9$

(3(5)+5)mod 11 = 9

0	1	2	3	4	5	6	7	8	9	10
13	94				44			12	16	20
	39				88			23	5	
					11					

11. (10 points) R-9.8 p. 417

What is the result of the previous exercise, assuming collisions are handled by linear probing?
(3(12)+5)mod 11 = 8
(3(44)+5)mod 11 = 5
(3(13)+5)mod 11 = 0
(3(88)+5)mod 11 = 5->6
(3(23)+5)mod 11 = 8->9
(3(94)+5)mod 11 = 1
(3(11)+5)mod 11 = 5->7
(3(39)+5)mod 11 = 1->2
(3(20)+5)mod 11 = 10
(3(16)+5)mod 11 = 9->3
(3(5)+5)mod 11 = 9->4

0	1	2	3	4	5	6	7	8	9	10
13	94	39	16	5	44	88	11	12	23	20

12. (10 points) R-9.10 p. 417

What is the result of Exercise R-9.7, when collisions are handled by double hashing using the secondary hash function $h_s(k) = 7 - (k \bmod 7)$?
(3(12)+5)mod 11 = 8
(3(44)+5)mod 11 = 5
(3(13)+5)mod 11 = 0
(3(88)+5)mod 11 = 5->7-(88 mod7) = 3->(5 + 3)mod 11 = 8->(5 + 2(3))mod 11 = 0->(5 + 3(3))mod 11 = 3
(3(23)+5)mod 11 = 8->7-(23 mod7) = 5->(8 + 5)mod 11 = 2
(3(94)+5)mod 11 = 1
(3(11)+5)mod 11 = 5->7-(11 mod7) = 3->(5 + 3)mod 11 = 8->(5 + 2(3))mod 11 = 0->(5 + 3(3))mod 11 = 3->(5 + 4(3))mod 11 = 6
(3(39)+5)mod 11 = 1->7-(39 mod7) = 3->(1 + 3)mod 11 = 4
(3(20)+5)mod 11 = 10
(3(16)+5)mod 11 = 9
(3(5)+5)mod 11 = 9->Last available slot.

0	1	2	3	4	5	6	7	8	9	10
13	94	23	88	39	44	11	5	12	16	20