CSCE 221 Cover Page

Homework Assignment #2

First Name: Kevin Last Name: Chou UIN: 224000120

User Name: k11 E-mail address: k11@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				
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 Kevin Chou

Date 10/18/15

Homework 2 due October 18 at 11:59 pm.

1. (20 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.

$A(n)=1+A_{n-1}$

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

$A(n)=n$

$O(A(n)=n$

2. (20 points) Write a recursive function that finds the maximum value in an array of int values without using any loops.

```
int max(int* array, int max=0, int arraySize) {

if(*array>max)

        max=*array;

if(arraySize>1)

        max(array[1], max, arraySize-1);

return max;

}
```

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

$A(n)=1+A_{n-1}$

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

$A(n)=n$

$O(A(n)=n$

3. (10 points) What data structure is the most suitable to determine if a string is a palindrome? A string is a palindrome if it is equal to its reverse. For example, "racecar" and "so many dynamos" are palindromes (spaces are removed from many word strings). Justify your answer. Use Big-O notation to classify the running time of your algorithm.

Stack because of the first-in last-out behavior. Pushing in letters of a word into a stack and the popping them in the reverse order forms the word pushed in but in reverse. The 2 words can then be compared to see if they match and if they do the word is palindrome.

4. (10 points) Solve C-5.2 on p. 224

> You pop all of the elements in stack S into queue Q until you find element X. When element X is found, you deque Q and push the elements back into stack S. The number of elements that were taken out will be the on top of the remaining elements in the stack and the stack will be out of order. So you pop those same elements from the stack into queue Q and then when you deque Q and push the elements will be in order because of the queue's first-in, first-out behavior.

5. (20 points) What is the amortized cost of the stack push operation when the additional stack-array memory is allocated by each of these two strategies? Do calculations to support your answer.

(a) Doubling strategy – double the size of the stack-array memory if more memory is needed.

O(1)

Doubling a stack-array guarantees enough space for the next push operation.

(c) Incremental strategy – increase the size of the stack-array by a positive constant c if more memory is needed.

$\Theta(n^2)$
Every time a push operation is used, the array needs to be incremented. Pushing n elements would require n increments giving it a $\Theta(n^2)$ amortized cost.

6. (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?

Queue 1 = q1

Queue 2 = q2

while (q1 size > 1)

deque q1 except last element into q2

return last element

deque q2 into q1

Runtime of push is 1 O(n)=1

Runtime of pop is 2n O(n)=n

7. (10 points) Solve C-5.8 on p. 224

You push the numbers until an operand is found and then you pop the 2 previous elements before hitting the operand. The operand found will be the one used for the 2 numbers that were popped. The resulting number is then pushed back into the stack. The process is repeated until another operand is hit or there are no more elements.

8. (20 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.

Best case: 2 1 4 3 the pivot will always be the lower bound for each partition

Worst case: 1 2 3 4 the pivot will always be the next element

Average case: 4 3 1 2 the pivot will be dependent on how many elements will be less the right partition

(b) Write a recursive relation for running time function and its solution for each case.

Best case: $T(n)=2*(T_{n/2})+n$

Solution: $T(n)=n\log n$

Worst case: $T(n)=n*(T_{n-1})+2$

Solution: $T(n)=((n-1)*(n-2))/2$

Average case: $T(n)=n\log n$

9. (15 points) Consider the merge sort algorithm.

(a) Write a recurrence relation for running time function for the merge sort.

$T(n) = 2T(n/2)+n$

(b) Use two methods to solve the recurrence relation.

Master theorem:

$a=2,\ b^d=2,\ a=b^d,$

$O(n^{\log_b d} \log n)$

Iterative method:

$T(n)=2T(n/2)+0$

$T(n-1)=2T(n/4)+n$

$T(n-2)=2/(2T(n/8))+2n$

$T(n-(n-k)) = K(T(n/k)+n\log k - k+1)$

$O(n\log n)$

(c) What is the best, worst and average running time of the merge sort algorithm? Justify your answer.

$O(n\log n)$