**Name:**

**Regis University CC&IS**

**CS324 Algorithms & Analysis**

**Homework 1**

**Assignment**Place your answers directly under each question below -- do not delete the questions.

***Part 1: Algorithm Complexity Analysis***

1. Give the big-Oh characterization, in terms of **n**, of the running time of the pseudocode loops shown below: (*3 pts each*)
2. x 🡨 1

for i 🡨 1 to 8n do

x 🡨 x \* i  
O(**n**)

1. b 🡨 1

while n > 0 do

b 🡨 b + (5 \* n)

n 🡨 n / 2  
O(**n**)

1. s 🡨 0

for i 🡨 1 to 2n do

for j 🡨 1 to i do

s 🡨 s + i  
O(**n2**)

1. k 🡨 0

for i 🡨 1 to n2 do

for j 🡨 1 to 5n do

k 🡨 k + i

O(**n3**)

1. Complete problem C1.5 on page 45 of your textbook (*6 pts*)

Show ALL math conversion steps.

1. Given the following functions, determine the big-O notation for each (3 *pts each*)

Hint: Use the rules given in the textbook.

1. f(n) = 500n2 + 50n3 + 5n

O(**n3**)

1. f(n) = 5n log n + 888n

O(n log n)

1. f(n) = 33n2 + 5n2 + 1000n2

O(n2)

1. f(n) = 8n3 + 2n + 44n2

O(2n)

1. Consider the following algorithm (shown with line numbers) that finds the maximum element in an array of positive integers.

**1: int findMaxElement(int nums[], int N) {**

**2: int i, maxNum;**

**3: maxNum = −1;**

**4: for (i = 0; i < N; i++)**

**5: if (nums[i] > maxNum)**

**6: maxNum = nums[i];**

**7: return maxNum;**

**8: }**

You can assume all elements are unique and that each occurs with equal probability.

1. Best case
   1. Describe the best case **data** for this algorithm. *(2 pts)*
   2. Then, for this best case, state how often is line 6 executed. *(2 pts)*
2. Worst case
   1. Describe the worst case **data** for this algorithm. *(2 pts)*
   2. For this worst case, state how often is line 6 executed. *(2 pts)*
3. Suppose that for the worst case, given input size **n**:

Algorithm 1 does **f(n) = n2 + 4n** steps

Algorithm 2 does **f(n) = 45n + 10** steps

For what input sizes is algorithm 1 faster than algorithm 2, in the worst case? (*4 pts)*

1. Given the following ordered array: (*5 pts each*)

|  |  |
| --- | --- |
| [0] | 2 |
| [1] | 4 |
| [2] | 7 |
| [3] | 9 |
| [4] | 22 |
| [5] | 36 |
| [6] | 44 |
| [7] | 67 |
| [8] | 68 |
| [9] | 71 |
| [10] | 83 |
| [11] | 85 |
| [12] | 88 |
| [13] | 90 |
| [14] | 94 |
| [15] | 99 |

1. Using the binary search algorithm to search for the target value **88**

For each pass, list the high and low index, and the calculated middle index.

Finally, state what causes the search to stop, and the result of the search.

1. Using the binary search algorithm to search for the target value **60**

For each pass, list the high and low index, and the calculated middle index.

Finally, state what causes the search to stop, and the result of the search.

1. Given the following recursive method:

public static int recurMethod (int num1, int num2) {

if (num1 < num2)

return num2;

else

return (num1 – num2) + recurMethod(num1 – 3, num2 + 5);

}

If the initial call is:

**int result = recurMethod (50, 30);**

1. List each recursive call going forward (*3 pts*)
2. List the value returned from each call, and the final result. (*5 pts*)
3. Assume you have a 20-element hash table that uses the hash function:   
        **H(key) = key mod 20**  
   and ***quadratic probing*** to resolve collisions.   
   If the following keys were inserted into the hash table, in the order shown: **39, 59, 23, 40, 79**   
   where will each be placed (i.e. at what index)? SHOW how you got your answers. *(8 pts)*
4. Begin with an empty binary search tree. (5 *pts*)

Insert the following keys, in the given order:

43, 30, 40, 12, 65, 86, 71, 33, 95, 88, 35  
Draw the resulting tree.

1. Starting with the binary search tree shown below. (*5 pts each*)

# 

1. List the order the nodes will be visited, using pre-order traversal.
2. List the order the nodes will be visited, using post-order traversal.
3. Deletion algorithms either use the **left** subtree or **right** subtree to find the node to replace the deleted node (must use ***same choice for all deletions***).   
   ***State which subtree you will use***.

Then show the resulting tree, after delete nodes 55, 52, 19, 77, 45, in that order.

1. Complete problem **A1.8** on page 49 of your textbook (*8 pts*)

Present the algorithm in pseudocode and then explain the runtime of the algorithm.

Do not use any built-in code library methods in your solution.

1. Compare and contrast some of the previously studied data structures:

Hash tables

Binary trees

Discuss advantages and disadvantages of each, in terms of the time it takes to **add** and **delete** elements within the data structure, and the **memory required** for the data structure. (*4 pts*)

**Submission**

This homework assignment is due by midnight of the date listed on the **Course Assignments by Week** page of the Content.

* Before you submit your homework, append your last name to the front of the Word doc filename. For examples: **SmithCS324-Hwk1.docx**
* Submit your **.docx** file to the **Hwk Assn 1** Submission Folder (located under **Assignments** tab in online course

***WARNING:****Homework submitted more than* ***3 days*** *past the due date will* ***not*** *be accepted,  
and will receive a grade of 0.*