Faculty of Science

Course:	CSCI 3070U: Analysis and Design of Algorithms
Assignment:	#1
Topic:	Chapters 1-4

Note: This assignment is to be completed individually. This means you should not work in teams. These restrictions are in place because the idea of this assignment is to prepare you to do well on the upcoming midterm, which will include questions related to this assignment.

Part 1

Find upper bounds, O(n) notation, for the following recurrences. Use the Equation editor in MS Word in order to ensure that your answers are readable, except for recursion tree where I'd recommend you create an image like in the week 2 slides. Show your work. Be sure to solve at least one of the problems using each of recursion tree, substitution, and master method.

a. $T(n) = 2T(n/2) + n/\log n$
b. $T(n) = 7T(n/2) + n^2$
c. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$
d. $T(n) = 2T(n/4) + \sqrt{n}$

Part 2

Rank the following functions by their order of growth. Be sure to put similar functions into the same category (f and g are in the same category iff f(n) = Θ(g(n))). The result should be a table. The first column should be the category (and will span multiple answers if there is more than one function in that category). The second column should be the function itself. You do not need to justify your answers for this question.

n^2	$3n^2+7n+15$
$2^{\log_2 n}$	$\log_{10} n$
$n^3-\log n$	4^n
$n^{71}+5^n+17n$	$18n$
2^n	$3 \log_2 n$
$\sqrt{n^3}$	$n!$
$\frac{3}{4}n^4$	n^3

Part 3

Using divide and conquer, create an algorithm to solve the problem that follows. You can use C, C++, Java, or Python to solve this problem (ask the instructor if you have another programming language in mind). Include the asymptotic upper bound for your algorithm in your response (including the cost of subdivision and combining the results).

> *You have a long string containing many characters (such as this paragraph), and you want to search for a substring within this string. For example, one may want to search for "characters" or "want to" or "bstring wi" or "language". All but the last example should be found.*

Keep in mind that if you use divide and conquer to solve this problem there is one complication. The string to be found could be split between two of the sub-problems (assuming your algorithm divides the string into two smaller strings). You'll need to handle that case as well.

Note: *This is a challenging problem. Just do your best, and partial marks will be given for using the correct divide and conquer strategy.*

Part 4

For this part of the assignment, you will implement the heap data structure, and use it to implement a heap sort in Java, C, C++, Python (or another approved programming language).

The heap data structure will need at a minimum the 5 operations discussed during the lectures, as shown below. You can rename these operations away from the naming conventions of the textbook, as long as it is clear what each of them does.

- `BUILD-MAX-HEAP`: Takes an arbitrary array and builds it into a max heap
- `MAX-HEAPIFY`: Takes an almost-heap with one violation, and fixes the violation
- `HEAP-MAXIMUM`: Returns the largest element in the max heap
- `HEAP-EXTRACT-MAX`: Removes and returns the largest element in the max heap
- `MAX-HEAP-INSERT`: Inserts a new element into the heap, preserving the heap property

In addition to these operations, you should also implement two display methods, for testing purposes.

- `printAsArray`: Prints the array representation (e.g. [16,14,10,8,7,3,9,1,4,2])
- `printAsTree`: Prints the heap as a sideways tree, as shown below:

```
      9
   10
      3
16
      7
         2
   14
         4
      8
         1
```

Hint: For `printAsTree()`, you'll find it easier to write the function recursively including a depth argument (which indicates how far to the right to indent the output). Recursion will be used to print the left and right subtrees (with a incremented depth).

The implementation of `hashsort()` will be a function that takes an arbitrary array, and sorts it using a heap. Be sure to include testing code, where you create a heap from an arbitrary array, use `buildheap()`, and then heapsort the array (printing the array before and after), as well as testing the other operations mentioned above.

How to Submit

Put your answers to part 1 and 2 (as well as the upper bound for part 3) into a Word or PDF file, called `Assignment1.docx` or `Assignment1.pdf`. Put your program for parts 3 and 4 into a file appropriately named (e.g. `DivideAndConquer.java`, `divide_and_conquer.py` and `Heap.java`, `heap.py`). Zip both of these files into a file named using the pattern `Asmt1_FirstNameLastName_StudentNumber.zip` (e.g. `Asmt1_RandyFortier_100539147.zip`), and submit this file to the Blackboard drop box for this assignment.