CS 692 Algorithms Capstone Exam
Spring 2024

Please read the following instructions before you start the exam. Please avoid asking a question that is addressed below:

1) You can write your answers **on both sides of the pages**. If you run out of space as you answer the questions on the front pages, just continue your answer on the back of the page. Please do not ask the instructor whether you can write on the back side or not again. The answer is YES YOU CAN.

2) You may use the **last paper** sheet (front and back) as the scratch paper. It is also marked as scratch paper on both sides. Please note that the instructor will NOT grade the scratch paper. The scratch sheet stays with the instructor. Please do not take that apart.

3) Do NOT disjoint the exam sheets. If at any point the papers got disjointed, raise your hand and request for stapling them immediately.

4) You are **NOT** allowed to use Calculator.

5) You are NOT allowed to have your cell phone, e-watch, or other electronic devices nearby like on the desk or inside your pocket. They should be turned off and stay inside your bag /backpack.

6) Exam duration: from 3:00 to 4:30 pm.

Please indicate the questions you have completed for grading. If not, we will assume that the first two questions you attempted are to be graded.

☐ **Question 1**

☐ **Question 2**

☐ **Question 3**

Full name (print): ______________________ **Net ID:**__________

Choose any 2 of the 3 problems.
If you answer all three questions, only questions 1 and 2 will be graded.

Full name: ________________ Net ID:________________

Question 1) (20 points)

A) (**10 points**) Consider the following recurrence relations and solve them to come up with a precise function of n in closed form (that means you should resolve all ∑'s, recursive calls of the function T, etc.). An asymptotic answer (such as one that uses Big-O, Theta, and similar) is NOT acceptable. Justify your solution and show all your work.

$T(n) = T(n/2) + 2\log_2(n)$, where $T(1) = 1$ and $n = 2^k$ for a non-negative integer k.

B) (**10 points**) Count the precise number of "fundamental/basic operations" executed in the code below. Your answer should be a function of n in closed form. Note that "closed form" means that you must resolve all ∑'s, recursive relations, etc. An asymptotic answer (such as one that uses Big-O, Theta, and similar) is not acceptable. Justify your solution and show all your work.

```
for (int k = 2; k <= n; k++)
{
        Perform 1 fundamental/basic operation;
        for (int j = k; j < n; j++)
                Perform 1 fundamental/basic operation;
         //endfor j
 }//endfor k
```

Full name: ____________________ Net ID: ____________________

Full name: ________________ Net ID:________________

Question 2) (20 points)

A) **(9 points)** Let two functions f(n) and g(n) reflect the total number of basic operations in two algorithms respectively. Which of the following seven statements correctly describes the relationship between the functions f(n) and g(n) where the limitation of f(n) over g(n) is defined in 1)-3) below? Note that more than one of the seven statements may be correct for each part. The 3 points for each part will be evenly distributed among all correct statements for that part, for example, if there are supposed to be 6 correct statements for part 1), then each correct answered statement will get 0.5 points; if there are supposed to be 7 correct statements for 1), then one statement will be 0.42 points, and each of the rest 6 correct statements will get 0.43 points (0.42 + 0.43*6= 3).

DO NOT GUESS: positive points will be given for a correct statement; -1 point will be given for each incorrect statement.

$f \in \Omega(g)$ $f \in o(g)$ $f \in O(g)$ $g \in \Omega(f)$ $g \in o(f)$ $g \in O(f)$ $f \in \theta(g)$

1) **(3 points)** $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

2) **(3 points)** $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, where $0 < c < \infty$

3) **(3 points)** $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

B) (**11 points**) Consider the following algorithm for converting an array *a[1..n]* of size n into a binary **max-heap**:

```
for (int i = 2; i <= n; i ++)
    Insert( a[i] into heap a[1..i-1])
```

Apply this algorithm to the following array to convert it into a binary max-heap, by drawing both the array *a* and the tree of the heap *a[1..i-1]* after each step.

Array *a* :

50	60	90	80	20	10

Note: The question is about binary **max-heap**, a binary min-heap will not get any points.

Full name: ____________________ Net ID:____________________

Full name: ________________ Net ID:________________

Question 3) (C/C++ coding question) (20 points)

Given two **balanced binary search trees** as T_1 and T_2, where T_1 has *n* nodes and T_2 has *m* nodes, the goal is to merge these two trees into a **new balanced binary search** tree T_3 constructed out of values present in T_1 and T_2. You can assume that each node will be assigned different and distinct integer labels, meaning no two nodes in trees T_1 and T_2 share the same label.

One way to do this is to first convert each binary search tree into a sorted array or list by doing an in-order traverse of the tree, then merge the sorted two arrays or lists into one sorted array or list, and finally convert the new sorted array back to a balanced binary search tree.

Part a) (2 points) Declare the node structure of your binary search tree in C++. Only node structure, NOT a class for the binary search tree with search, insert or delete etc. operations.

Part b) (8 points) Write a C++ function that merges two sorted arrays of distinct integers with size *n* and *m* respectively into one sorted array and returns the new sorted array. The time complexity of your function should be in $O(n+m)$.

Part c) (10 points) Write a recursive function in C++ that converts a sorted array of $n + m$ distinct integers to a balanced binary search tree with nodes that you declared in **Part a)**, the integers of the sorted array will be the node values. Give the time complexity of this recursive function using Big-O notation.

For **Part a), Part b) and Part c)**,

Note #1: Only C++ programming is accepted; any other language will receive no points.

Note #2: You are not permitted to utilize pre-existing (built-in) routines that perform the specific tasks we have asked you to do. You are expected to implement these operations from scratch in your code.

Note #3: Ensure that your code output stays within the requirements: such as sorted, balanced, and time complexity requirement, if given.

Note #4: You are not required to give test cases for your functions, but if you do, make sure you give distinct and unique label values for the arrays or the trees.