CS 692 Algorithms Capstone Exam, Fall 2025

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)
Part A) (14 points) Consider the following recurrence relations and solve each to come up with a precise function of n in closed form (that means you should resolve all sigmas, recursive calls of the function T, etc.). **An asymptotic answer is NOT acceptable**. Justify your solution and **show all your work.**

a) T(n) = T(n-1) + 5, where $T(1) = 1$

b) $T(n) = 2\ T(\frac{n}{2}) + n$, where $T(1) = 1$ and $n = 2^k$ for a non-negative integer k

Part B) (6 points) Consider the following recurrence relation. Determine its asymptotic complexity in Big-O notation. You may apply the Master Theorem or any other valid method of your choice. **Clearly explain your reasoning and provide the final result in $O()$ notation.**

a) $T(n) = 4T(\frac{n}{2}) + 20\ n^2$, where $T(1) = 1$

Full name: _____ Net ID:_____

Question 2)

Part A) (12 points) For each *function* f(n) *below*, give an *asymptotic upper bound* using "**Big-O()**". You should give the **tightest** bound possible. **No need to justify your answer**.

(1) $f(n)= n^2 + n \log n$

(2) $f(n)=n^{100} + n!$

(3)

$$f(n)=\begin{cases} n^3 + 20n, & n \leq 12 \\ 3n + 5, & n > 12 \ \ and\ n\ is\ odd \\ 45n^2 + 20, & n > 12 \ \ and \ \ n\ is\ even \end{cases}$$

(4) $f(n)= n^n + 2^n + 10^{10}$

Part B) (8 points)

Let f(n) and g(n) be two functions representing algorithmic runtimes. The growth rate of f(n) compared to g(n) can be analyzed using the limit below. Analyze the behavior of the limit

$$\lim_{n\to\infty} \frac{f(n)}{g(n)}$$

in each of the following cases.

a) $f(n) \in o(g(n))$ (little-o of g(n))
b) $f(n) \in O(g(n))$ (Big-O of g(n))

Then, compare your findings for case (a) and case (b), discussing their similarities and differences.

Full name: _____ Net ID:_____

Question 3) (20 points)

1) **(4 points)** Define a stack data structure and specify its properties.
2) **(16 points)** Implement (in C/C++ only) a **stack** of integers using **a singly linked list**. **Declare the data structure** and provide code for the following operations:
   a) **empty_check:** this operation checks whether the stack is empty.
   b) **push**: inserts an element into the stack.
   c) **pop**: removes and returns the top element of the stack.
   d) **find_max**: returns the maximum value in the stack without altering its contents. All elements must remain in the stack in their original order, and the operation must not modify the stack in any way.

Important Notes:

1. Only C or C++ implementations will be accepted. Submissions in any other language will receive no credit.
2. You must use a **singly linked list** with **pointers** for your implementation.
3. You are not permitted to use built-in libraries or pre-existing routines that perform stack operations. All functionalities must be implemented from scratch.