Duration: **60 minutes**
Aids Allowed: **one *single*-sided *hand*written 8.5"×11" aid sheet**

Student Number: |_|_|_|_|_|_|_|_|_|_|_|

Family Name(s): _____

Given Name(s): _____

Lecture Section: ☐ L0101 (A. Farzan) ☐ L0201 (F. Pitt)

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This term test consists of 4 questions on 10 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete, fill in the identification section above, and write your name on the back of the last page.*

Answer each question directly on the test paper, in the space provided, and use one of the "blank" pages for rough work. If you need more space for one of your solutions, use a "blank" page and *indicate clearly the part of your work that should be marked.*

In your answers, you may use without proof any theorem or result covered in lectures, tutorials, homework, tests, or the textbook, as long as you give a clear statement of the result(s)/theorem(s) you are using. You must justify all other facts required for your solutions.

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do — part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

MARKING GUIDE

1: _____/ 4

2: _____/ 8

3: _____/ 6

4: _____/ 8

BONUS MARKS: _____/ 3

TOTAL: _____/26

Good Luck!

Question 1. [4 MARKS]

Consider the following algorithm:

```
EXP(a, b, p):      # Computes and returns a^b mod p.
1.  if b = 0:  return 1
2.  if b mod 2 = 1:      # i.e., b is odd
3.      x = EXP(a, (b − 1)/2, p)
4.      return (x × x × a) mod p
    else:
5.      x = EXP(a, b/2, p)
6.      return (x × x) mod p
```

Give a recurrence relation for the **exact** number of multiplication operations ("×") performed by this algorithm on inputs of size n, in the worst-case. Explain what you are doing, *i.e.*, state what n is equal to (in terms of the algorithm's variables), and what part of the algorithm is represented by each term in your recurrence (use the line numbers to the left of each line for easy reference). **Do *NOT* try to give a closed-form solution for your recurrence!** All we want is the recurrence itself.

Question 2. [8 MARKS]

Consider the following recurrence relation:

$$T(1) = 2,$$
$$T(n) = T(\lceil n/3 \rceil) + g(n) \qquad \text{for } n > 1.$$

Give a **tight** bound on the closed-form solution for $T(n)$, for each function $g(n)$ on the next page. Show your work, *i.e.*, describe clearly how to apply the Master Theorem when it is possible, and when it is not, explain how you obtained your closed-form — **but do *NOT* write a proof**!

Question 2. (CONTINUED)

Part (a) [2 MARKS]

$g(n) = 4n$

Part (b) [2 MARKS]

$g(n) = 5$

Part (c) [4 MARKS]

$g(n) = 6^n$ (HINT: $5 + 5 \cdot 6 + 5 \cdot 6^2 + \cdots + 5 \cdot 6^n = 6^{n+1} - 1$.)

WARNING: This may be a bit long. Don't spend too much time on it at first — plan your time carefully!

Question 3. [6 MARKS]

Suppose that we want to find, at the same time, both the minimum *and* the maximum elements in an unsorted list. This can be done with two loops: one to find the minimum (using exactly $n-1$ comparisons between list elements), and the other to find the maximum (using $n-2$ comparisons between list elements, because we can skip the comparison with the minimum element). This "naive algorithm" makes a total of $2n-3$ comparisons between list elements, and we would like to do better.

Write a divide-and-conquer algorithm MINMAX that returns, as a pair, the minimum *and* maximum elements in a list. *You do* ***NOT*** *have to show that your algorithm makes fewer than* $2n-3$ *comparisons between list elements — for this question, just write a correct algorithm.*

HINT: This is simple: don't overthink it! Include two base cases: one for $n=1$ and one for $n=2$.

Question 4. [8 marks]

State precise preconditions and postconditions for your algorithm MinMax from Question 3, then write a detailed proof that your algorithm is correct.

Note: You may complete this part and get most of the marks, even if your algorithm is not actually correct, as long as it is clear that you understand how to prove the correctness of your algorithm.

Question 4. (CONTINUED)

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Bonus. [3 marks]

WARNING! This question is difficult and will be marked harshly: credit will be given only for making *significant* progress toward a correct answer. Please attempt this bonus only *after* you have completed the rest of the test.

Prove that your algorithm MinMax from Question 3 makes *fewer* comparisons between list elements than the naive algorithm described in Question 3. Explain what you are doing and show your work.

On this page, please write nothing except your name.

Family Name(s): _____

Given Name(s): _____

Total Marks = 26