CSC 236H5 F 2017 Test 2
Duration — 50 minutes
Aids allowed: 8.5x11 two-sided sheet

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

Last Name: _____ **First Name:** _____

☐ Lecture Section: L0101 Instructor: Dan Zingaro (13:00-15:00)
☐ Lecture Section: L0102 Instructor: Dan Zingaro (15:00-17:00)
☐ Lecture Section: L0103 Instructor: Sadia Sharmin (18:00-20:00)

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)
Good Luck! Take a deep breath. You got this! :) This is your chance to show us what you've learned. Remember, a number does not define you.

This test consists of 4 questions on 8 pages (including this page). *When you receive the signal to start, please make sure that your copy is complete.*
If you use any space for rough work, indicate clearly what you want marked.

1: _____/ 4
2: _____/ 5
3: _____/ 8
4: _____/ 3

TOTAL: _____/20

Total Pages = 8

Question 1. [4 MARKS]

Part (a) [2 MARKS]

Choose the worst-case runtime recurrence and Theta bound for the following D&C algorithm. A user would call the function with `select_number(A, 0, len(A)-1)`.

```
def select_number(A, low, high):
    if low == high:
        return A[low]
    mid = (low+high) // 2
    if A[mid] < A[mid+1]:
        return select_number(A, mid+1, high)
    else:
        return select_number(A, low, mid)
```

A $T(n) = 2T(n/2) + d; \Theta(n)$

B $T(n) = 2T(n/2) + n; \Theta(logn)$

C $T(n) = T(n/2) + d; \Theta(n)$

D $T(n) = T(n/2) + d; \Theta(logn)$

E $T(n) = T(n/2) + n; \Theta(logn)$

Part (b) [2 MARKS]

Consider function $T(n) = T(n/2) + T(n/4) + 8n$.

A The Master Theorem applies; $T(n) = \Omega(n)$

B The Master Theorem applies; $T(n) = O(nlogn)$

C The Master Theorem does **not** apply directly, but we can still use it to argue that $T(n) = \Omega(n)$

D The Master Theorem does **not** apply directly, but we can still use it to argue that $T(n) = O(nlogn)$

E More than one of the above

Question 2. [5 MARKS]

Consider the following recurrence that results from some unspecified divide-and-conquer algorithm, where a is a positive constant:

$$T(n) = \begin{cases} a, & \text{if } n = 1 \\ a, & \text{if } n = 2 \\ T(n/2) + T(n/3) + 1, & \text{if } n > 2 \end{cases}$$

Use a form of **induction** to prove that $T(n) = O(n)$. Do **NOT** use the substitution method.

Question 3. [8 MARKS]

Consider the following function.

```
def power(a, b):
    '''Given two natural numbers a and b, return the value of a to the power of b.'''

    if b == 0:
        return 1
    x = power(a, b//2)
    if b % 2 == 1:
        return x * x * a
    else:
        return x * x
```

Part (a) [1 MARK]

State the precondition and postcondition for the above function.

Part (b) [2 MARKS]

Which of the following can we use to measure the size of recursive calls? **Circle *all* that would work**.

1. `a`
2. `b`
3. `x`
4. `a+b`

Part (c) [5 MARKS]

Prove correctness of the above recursive function.

[You may use this page to continue your answer for the previous question.]

Question 4. [3 MARKS]

Consider the following function.

```
def mystery(x):
    '''
    Pre: x is a natural number >= 0
    Post: x == 0
    '''
    j = 0
    while x > 0:
        if j == 0:
            x = x - 1
            j = 1
        else:
            j = 0
    return x
```

Part (a) [1 MARK]

State (but **do not** prove) an invariant that could help us show that this function satisfies the postcondition.

Part (b) [2 MARKS]

State (but **do not** prove) a suitable variant that could be used to show that the loop terminates.

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]

[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]