MA/CSSE 473 – Design and Analysis of Algorithms **In-class Quiz Day 02**

Name:_____**Solution**_____ Score:____/ 18 circle your Section # 01(3rd) 02 (4th)

1. (2) What is your group's definition of "Algorithm"? Who is in your group?

   Any answer that seems to be answering this question is okay

2. (2) What is the class's composite definition of "Algorithm"?

   Any answer that seems to be answering this question is okay

3. (2) Write down your group's **Session→ Day of the Week** algorithm.

   Here is one possible solution. They may also do it iteratively:

```
def matrix_power(m, n):
    result = identity_matrix
    power = m
    while n > 0:
        if n % 2 == 1:
            result = matrix_multiply(result, power)
        power = matrix_multiply(power, power)
        n = n //2
    return result
```

4. (1) How do we know that the recursive computation of the n^{th} Fibonacci number, F(n), needs at least F(n) steps?

   Because in each part of the recurrence relation, the right side is bigger than the corresponding right side for the Fibonacci recurrence..

5. (1) How many additions _____ and multiplications ____ are required in order to multiply two 2x2 integer matrices?
   In the straightforward algorithm, 84 additions, 8 multiplications.

6. (3) With your group, write an efficient algorithm for finding the n^{th} power of a 2x2 matrix, assuming that you already have a constant-time algorithm for multiplying two matrices at a time.

```
def matrix_power(m, n):
    result = identity_matrix
    power = m
    while n > 0:
        if n % 2 == 1:
            result = matrix_multiply(result, power)
        power = matrix_multiply(power, power)
        n = n //2
    return result
```

(Quiz continues on the back of this page)

7. (2) How many matrix multiplications are needed in order to compute the nth Fibonacci number? Explain briefly.

   Any answer that is $\Theta(\log_2 n)$ is okay.
   Reason: It's the maximum number of recursive calls in the above algorithm

8. Why can't we just use the formula (seen in CSSE 230) that uses powers of the golden ratio?

Calculating powers of floating-point numbers is expensive, especially if we use a (not supported directly by hardware) representation of those number that stores enough digits of each number so we get accurate results for large Fibonacci numbers. Student answers do not need to be this detailed.

9. (1) When we add three 1-digit integers, how many digits can be in the answer? Is this independent of the base (i.e, the same for decimal, binary, hexadecimal, etc.).

   At most two digits. It is independent of the base.

10. (1) How does the previous question apply to the analysis of the addition of two n-bit non-negative integers?

    In doing a bitwise add, there can be at most three bits to add at each stage (the corresponding bits from the two numbers plus a carry bit, and the carry can never be larger than a one-bit number. Thus the addition is O(n).

11. (1) Tell me about anything from today's lecture that you found confusing or feel that we need to spend more time on?. (or write N/A).

    Must have an answer; any answer is okay

12. (1) What questions do you have (from today's lecture, from the reading, or form the course in general).
    Must have an answer; any answer is okay