CS:3330 Quiz 1, Spring 2018 (Version 1)

Let us define a problem called ArithmeticSeries as follows:
ArithmeticSeries
Input: A positive integer n.
Output: The sum $1 + 2 + \cdots + n$.
So if the input is 5, then the output should be 15. Here is an algorithm that solves this problem.

```
function arithSer(n)
1.    s ← 0
2.    for i ← 1 to n do
3.          s ← s + i
4.    return s
```

1. What is the input size for the problem ArithmeticSeries?

   $\log_2 n$.
   Explanation: It takes roughly $\log_2 n$ bits to represent n and so that is the input size.

2. What is the number of basic computer steps used by the function `arithSer` to solve the problem ArithmeticSeries? Express your answer, first as a function of n and then as a function of the input size you came up with in (1). Show your work to get partial credit.

   $2n \log_2 n + n + 3$ basic computer steps. Let $s = \log_2 n$ denote the input size. Then, as a function of s, the number of basic computer steps is $s \cdot 2^{s+1} + 2^s + 3$.
   Explanation: Line 1 takes 1 basic computer step, Line 2 takes $(n + 1)$ basic computer steps, and Line 4 takes 1 basic computer step, for a total of $n + 3$ basic computer steps. Now let us focus on Line 3. First note that Line 3 is executed n times.

   The sum $1 + 2 + \cdots + n$ is at most n^2. Therefore the variable s can be stored in at most $\log_2(n^2) = 2 \log_2 n$ bits. Since $i \leq n$, when it is used in Line 3, the variable i can be stored in at most $\log_2 n$ bits (in Line 3). Therefore, in Line 3, we are adding a number made up of at most $2 \log_2 n$ bits and a number made up of at most $\log_2 n$ bits. By the grade-school addition algorithm, this takes $2 \log_2 n$ basic computer steps. Therefore, Line 3 takes at most $2n \log_2 n$ basic computer steps over all n iterationsof the for-loop.

   Thus the entire algorithm takes $2n \log_2 n + n + 3$ basic computer steps.

3. Would you say this is an *efficient* algorithm? (Yes/No)

   No.
   Explanation: As seen in the above answer, the number of basic computer steps performed by this algorithm is exponential in s, the input size.

4. From basic algebra, we know that

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

   If you simply used the formula $n(n + 1)/2$ to compute the sum $1 + 2 + \cdots + n$, how many basic computer steps would that take? Show your work to receive partial credit.

   $\log_2^2 n + 4 \log_2 n$
   Explanation: Since it takes $\log_2 n$ bits to represent n, using the grade-school multiplication algorithm takes $\log_2^2 n$ basic computer steps to compute n^2. It takes an additional $2 \log_2 n$ steps to compute $n^2 + n$ and an additional $2 \log_2 n$ steps to divide $n^2 + n$ by 2.

5. Would you say that your algorithm from (4) is *efficient*? (Yes/No)

   Yes.
   Explanation: The number of basic computer steps performed by the algorithm is now $s^2 + 4s$, a polynomial function of the input size.

CS:3330 Quiz 1, Spring 2017 (Version 2)

Let us define a problem called ModuloGeometricSeries as follows:
ModuloGeometricSeries
Input: Positive integers n and p.
Output: The sum $(1 + 2 + 2^2 + \cdots + 2^n) \bmod p$.
So if the input is $n = 4$ and $p = 7$, then the output should be 3 (check this!). Here is an algorithm that solves this problem.

```
function modGeomSer(n, p)
1.   s ← 0; c ← 1;
2.   for i ← 1 to n do
3.        s ← (s + c) mod p
4.        c ← (2 * c) mod p
5.   return s
```

Suppose that p is guaranteed to be small and it fits in a single word of memory. To be concrete, you can assume that 1 word equals 32 bits.

1. What is the input size for the problem moduloGeometricSeries?

   $\log_2 n + 32$.

2. What is the number of basic computer steps used by the function `modGeomSer` to solve the problem moduloGeometricSeries? Express your answer, first as a function of n and then as a function of the input size you came up with in (1). Show your work to get partial credit.

   $7n + 4$. Let $s = \log_2 n + 32$ denote the input size. Therefore, $n = 2^{s-32}$ and we can re-express $7n + 4$ as $7 \cdot 2^{s-32} + 4$.
   Explanation: Line 1 takes 2 basic computer steps and Line 2 takes $(n + 1)$ basic computer steps. Each execution of Line 3 takes 3 basic computer steps. This is because all the variables involved in the arithmetic expression in this line are no greater than p and therefore can fit in one word of memory. Line 3 is repeated n times for a total of $3n$ basic computer steps. Similarly, Line 4 also takes a total of $3n$ basic computer steps. Finally, Line 5 takes 1 basic computer step. Thus the total number of basic computer steps performed by the algorithm is $7n + 4$.

3. Would you say this is an *efficient* algorithm? (Yes/No)

   No.
   Explanation: As shown above, the number of basic computer steps the algorithm takes is an exponential function of s, the input size.

4. From basic algebra, we know that

   $$1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1.$$

   If you simply used the formula $(2^{n+1} - 1) \bmod p$ along with the "repeated squaring" idea to compute the sum $(1 + 2 + 2^2 + \cdots + 2^n) \bmod p$, how many basic computer steps would that take? Show your work to receive partial credit.

   $\log_2 n$.
   Explanation: Using the "repeated squaring" idea and properties of modular arithmetic, to compute $2^{n+1} \bmod p$, we compute $2^{(n+1)/2} \bmod p$ and then square that. Since we are squaring a number that can fit in one word of memory, the squaring operation takes one basic computer step. As seen in class, the recursive function that implements the "repeated squaring" idea takes $\log_2(n + 1) \approx \log_2 n$ squarings, i.e., (approximately) $\log_2 n$ basic computer steps.

5. Would you say that your algorithm from (4) is *efficient*? (Yes/No)

   Yes.
   Explanation: As shown above, the number of basic computer steps is simply $s - 32$, where s is the input size.
