P1

1. *What is gcd(x!+1,(x+1)!+1), where x > 0?*

   gcd(x!+1,(x+1)!+1)
   = gcd(x! + 1, (x+1)! + 1 - (x! + 1))
   = gcd(x! + 1, x!*x)
   x! + 1 cannot be divided by any integers in range [2, x],
   because, for any integer i in range [2, x],
   $\frac{\mathrm{x}!+1}{i} = \frac{\mathrm{x}!}{i} + \frac{1}{i}$, where $\frac{\mathrm{x}!}{\mathrm{i}}$ $must\ be\ integer\ while \frac{1}{i}\ must\ not$

   Thus, x! + 1 doesn't have prime factors smaller than or equal than x
   mean while x!*x only have prime factors smaller than or equal than x
   thus gcd(x! + 1, x!*x) = 1

2. *How many integers n : 1 <= n <= 100, are there such that (n+1)(n+3) is divisible by 7?*

   For (n + 1)(n + 3) to be divisible by 7, either n + 1 is divisible by 7 or n + 3 is divisible by 7 (n + 1 and n + 3 cannot be both divisible by 7 at the same time)

   1 <= n <= 100, then 2 <= n + 1<= 101, 4 <= n + 3 <= 103

   $\left\lfloor\frac{101}{7}\right\rfloor = \left\lfloor\frac{103}{7}\right\rfloor = 14$
   there are 14 numbers divisible by 7,
   the smallest is 7, the largest is 7 *14 = 98

   So n can be i*7 – 1 or i*7 -3, where i = 1, 2, … , 14
   In total, 2 * 14 = 28 choices. Therefore, there are 28 integers n, s.t. (n + 1)(n + 3) is divisible by 7

3. *What is (1! + 2! + 3! + 4! + ... + 1000!) % 10?*

   $$\begin{aligned}&(1! + 2! + 3! + \cdots + 1000!)\ \%\ 10\\ &= (1!\ \%\ 10 + 2!\ \%\ 10 + \cdots 1000!\ \%10)\ \%\ 10\\ &\begin{pmatrix} for\ any\ number\ x \geq 5, x!\ is\ divisble\ by\ 10, \\ because\ 2\ and\ 5\ are\ factors\ of\ x!, 2 * 5 = 10 \end{pmatrix}\\ &= (1!\ \%\ 10 + 2!\ \%10 + 3!\ \%\ 10 + 4\%10)\ \%\ 10\\ &= (1 + 2 + 6 + 4)\%\ 10\\ &= 3\end{aligned}$$

4. *Our prisons are too crowded, so we need to release some inmates. For a high-security prison, there are 1000 inmates and 1000 guards. Initially, all doors are locked. Beginning with the 1st guard, the ith guard switches the locked/unlocked state of every ith door. For example, the first guard would go through and unlock every door. Then the 2nd guard switches the (lock/unlock) state for every even door (effectively locking every even door while leaving every odd door unlocked). Then the 3rd guard switches the state for every 3rd door, unlocking the door if it is locked, or locking the door if it is unlocked. After the 1000th guard, how many doors are left unlocked?*

   The doors that are left unlocked state are toggled odd times each.

   ith door will be toggled even times if i has even number of factors; it will be toggled odd times if it has odd number of factors.

   For any number x by which i is divisible, i is also divisible by the quotient, i/x. So i's factors are in pairs, unless x = i/x, which gives odd number of factors.

   Therefore, only square numbers will have odd number factors. (square number doors will be left unlocked)

In range [1, 1000], there are $\lfloor\sqrt{1000}\rfloor = 31$ square numbers. Therefore, there are 31 doors left unlocked at the end.

P2

1. *Explain how, given the value of h(s1 s2 ... sk), you can update it in constant time to obtain h(s2 s3 ... s(k+1)).*

   We can calculate the hash value by the following O(1) operation:
   h(s2 s3 … s(k + 1)) = h(s1 s2 … sk) - s1 + s(k+1)

2. *If h(si s(i+1) ... s(i+k-1)) = h(T), why have we not necessarily found a match? How would we verify whether this is actually a match?*

   Due to hash collisions, when hash values are same the strings are not necessarily equal. For example, 'b' + 'c' = 'a' + 'd', but "bc" != "ad". We then need to compare si s(i+1) ... s(i+k-1) and T character by character

3. *We will assume the probability of a false positive (that is, finding h(si s(i+1) ... s(i+k-1)) = h(T) when it isn't a match) is smaller than 1/k. Explain how the algorithm sketched out in your previous answers obtains an average runtime of O(n)*

   For i = from 1 to (n - k), we check h(si s(i+1) ... s(i+k-1)) = h(T), and when hash values are equal, we check character by character. We continue to next string when there is a false positive, or stop when there is a real match. Runtime to check one false positive match is O(k) and probability of false positive is 1/k, therefore:
   1/k * O(k) * O(n) + (k - 1)/k *O(1)*O(n) = O(nk/k + n) = O(n)

P3

Prove: The probability that any hash bucket contains more than ln(m) items goes to 0 as m goes to infinity. (As a result, for large hash tables, all hash operations are very likely to run in time O(log m).)

Place m items in m hash buckets

Hash function maps each item independently to a uniformly random position between 0 and m-1.

one item mapping to bucket i, probability is 1/m

a set ln(m) items all mapping to bucket i, probability is $(\frac{1}{m})^{\ln(m)}$

total number of sets = (m choose ln(m)) = $\frac{m^{\ln(m)}}{\ln(m)!}$

Probability of at least one set ending up in bucket i is upper bounded by sum of the probabilities of each set ending up in bucket i
= $\frac{m^{\ln(m)}}{\ln(m)!} * \left(\frac{1}{m}\right)^{\ln(m)} = \frac{m^{\ln(m)}}{\ln(m)!} * \frac{1}{m^{\ln(m)}} = \frac{1}{\ln(m)!}$

Probability of at least one bucket has more than ln(m) is upper bound by sum of probabilities of each bucket has more than ln(m) items
= $m * \frac{1}{\ln(m)!} = \frac{m}{\ln(m)!}$

when m goes infinite, m grows linearly, ln(m) grows exponentially

$$\lim_{m\to\infty} \frac{m}{\ln(m)!} = 0$$