HKUST – Department of Computer Science and Engineering

COMP 2711: Discrete Math Tools for Computer Science

Spring 2021 Midterm Examination

Date: Monday, 8 April 2021 Time: 19:30–21:20

Problem 1: [15 pts] Determine the truth value of each of the following statements. The domain for all variables are all integers. Justify your answer.

(a) $\exists x \forall y\ (y = x^2 + 2x + 1)$

(b) $\forall y \exists x\ (y = x^2 + 2x + 1)$

(c) $\exists x \forall y \exists z\ (2y - z = 4x)$

(d) $\exists x \exists z \forall y\ (2y - z = 4x)$

(e) $\exists x \exists y\ ((x > 1 \wedge y > 1) \rightarrow (x \bmod y \geq y))$

Solution : (a) False. For every x, there is always an integer $y = x^2+2x+2 > x^2+2x+1$.

(b) False. Note that $x^2 + 2x + 1 = (x + 1)^2$. When $y = 3$, there is no integer x so that $(x + 1)^2 = 3$.

(c) True. Let $x = 0$. Then for any y, set $z = 2y$.

(d) False. For every x and z, there is an integer $y = 4|x| + |z| + 1$, and thus $2y = 8|x| + 2|z| + 2 > 4x + z$.

(e) True. When $x \leq 1$ or $y \leq 1$, we have $F \rightarrow (x \bmod y \geq y)) = T$.

Grading Scheme: 3 pts each. For each part, 1 point for True/False. The remaining points for justification.

Problem 2: [12 pts] Consider the following predicates.

$c(x)$: "x studied COMP2711".

$g(x)$: "x knows how to compute the gcd of two integers".

$h(x)$: "x can get a high paying job".

Prove the following statement using inference rules.

"Sarah, a student who studied COMP2711, knows how to compute the gcd of two integers. Everyone who knows how to compute the gcd of two integers can get a high-paying job. Therefore, at least one student who studied COMP2711 can get a high-paying job."

Please first translate the premises and conclusion into predicate logic sentences. Then show a step-by-step proof using inference rules. You don't have to write down the name of the rule used; instead, you can just write down from which statement(s) a new statement is derived. For example, instead of writing "(3): statement (Modus tollens using (1) and (2))", you can just write "(3): statement (from (1) and (2))".

Solution : We are given premises $c(\text{Sarah})$, $g(\text{Sarah})$, and $\forall x(g(x) \rightarrow h(x))$, and we want to conclude $\exists x(c(x) \wedge h(x))$.

Step	Reason
1. $\forall x(g(x) \to h(x))$	Premise
2. $g(\text{Sarah}) \to h(\text{Sarah})$	Universal instantiation using (1)
3. $g(\text{Sarah})$	Premise
4. $h(\text{Sarah})$	Modus ponens using (2) and (3)
5. $c(\text{Sarah})$	Premise
6. $c(\text{Sarah}) \wedge h(\text{Sarah})$	Conjunction using (4) and (5)
7. $\exists x(c(x) \wedge h(x))$	Existential generalization using (6)

Problem 3: [10 pts] Show that there is no largest prime number.

Solution : We prove this by contradiction.

Suppose there is a largest prime number n. Note that $n! + 1$ has no factors between 1 and n. Thus, if all primes were n or less, $n! + 1$ would have no prime factors, and it would be prime itself. Therefore, $n!+1$ is a prime number greater than n, which is a contradiction.

Problem 4: [8 pts] Let $\mathbb{N}$ be the set of all natural numbers.

(a) Show that $\{S \mid S \subseteq \mathbb{N}, |S| \text{ is finite}\}$ is countable.

(b) Is $\{S \mid S \subseteq \mathbb{N}, |S| \text{ is infinite}\}$ countable or uncountable? Justification is not necessary.

Solution : (a) Here is one enumeration method for $\{S \mid S \subseteq \mathbb{N}, |S| \text{ is finite}\}$. We first enumerate $\emptyset$. Then for $k = 0, 1, \ldots,$ we enumerate all $S \subseteq \mathbb{N}$ such that $\max S = k$. For each k, there are finitely many such S, and we can enumerate them in any order.

(b) Uncountable.

Grading Scheme: 6, 2.

Problem 5: [10 pts] Alice and Bob share a key using the Diffie-Hellman key exchange algorithm with $a = 17, p = 31, k_1 = 34$ and $k_2 = 41$. What is the shared key? Note that the shared key must be in Z_p. Use Fermat's Little Theorem and the repeated squaring method to simplify the calculation. Show all the computational steps.

Solution : The shared key is $a^{k_1 k_2} \bmod p = 17^{34\cdot 41} \bmod 31$.

By Fermat's Little Theorem, $17^{34\cdot41} \equiv 17^{34\cdot41 \bmod 30} \equiv 17^{4\cdot11 \bmod 30} \equiv 17^{14} \pmod{31}$.

We compute the shared key by repeated squaring method.

$$17^{2^1} \bmod 31 = 10$$
$$17^{2^2} \bmod 31 = 10^2 \bmod 31 = 7$$
$$17^{2^3} \bmod 31 = 7^2 \bmod 31 = 18$$

Note that $17^{14} = 17^{2^1+2^2+2^3}$. So,

$$\begin{aligned} 17^{14} &\equiv 17^{2^1} 17^{2^2} 17^{2^3} \\ &\equiv 10 \cdot 7 \cdot 18 \\ &\equiv 20 \pmod{31} \end{aligned}$$

Problem 6: [12 pts] Consider the RSA encryption with parameters $p = 443, q = 211$.

(a) List out all possible public keys (n, e) so that e in $[120, 130]$. Justification is not necessary.

(b) Suppose you select the smallest value e in your answer of (a) to be the public key (n, e). Compute the corresponding private key d. Show all your steps.

Solution : (a) $T = (p-1)(q-1) = 442 \cdot 210 = 2 \cdot 2 \cdot 3 \cdot 5 \cdot 7 \cdot 13 \cdot 17 = 92820$. We need the values e in $[120, 130]$ that are relatively prime to T. The values cannot contain any factor in $\{2, 3, 5, 7, 13, 17\}$. Therefore, 121 and 127 are the only possible values of e.

(b) The private key should satisfy $(ed) \bmod T = 1$. i.e. d is the multiplicative inverse of e in Z_T. Run the extended GCD algorithm to find d:

$$\begin{aligned}
92820 &= 121 \cdot 767 + 13 \\
121 &= 13 \cdot 9 + 4 \\
13 &= 4 \cdot 3 + 1 \\
4 &= 1 \cdot 4 + 0
\end{aligned}$$

Then,

$$\begin{aligned}
1 &= 13 - 4 \cdot 3 \\
&= 13 - (121 - 13 \cdot 9) \cdot 3 \\
&= 13 \cdot 28 - 121 \cdot 3 \\
&= (92820 - 121 \cdot 767) \cdot 28 - 121 \cdot 3 \\
&= 92820 \cdot 28 - 121 \cdot 21479
\end{aligned}$$

Thus, $d = -21479 \bmod 92820 = 71341$.

Grading Scheme: 4, 8

Problem 7: [10 pts] Show that $p^{q-1} + q^{p-1} \equiv 1 \pmod{pq}$ for any two distinct prime numbers p and q.

Solution : By Fermat's little theorem, $p^{q-1} \equiv 1 \pmod{q}$.
Clearly, $q^{p-1} \equiv 0 \pmod{q}$.
Therefore, $p^{q-1} + q^{p-1} \equiv 1 + 0 \equiv 1 \pmod{q}$.
Similarly, $p^{q-1} + q^{p-1} \equiv 0 + 1 \equiv 1 \pmod{p}$.
By the Chinese remainder theorem, we have $p^{q-1} + q^{p-1} \equiv 1 \pmod{pq}$.

Problem 8: [15 pts] Consider the following three options.

(i) $f(x) = \Theta(g(x))$.
(ii) $f(x) = O(g(x))$.
(iii) $f(x) = \Omega(g(x))$.

For each of the following f, g pairs, choose all options that apply from above. No justification is necessary. Note that it is possible that none of them apply.

(a) $f(x) = \sin(x)$, $g(x) = (\cos(x))^2$.
(b) $f(x) = 2\sin(x)$, $g(x) = |\cos(x)| + 0.5$.
(c) $f(x) = 0.5x$, $g(x) = x|\cos(x)|$.
(d) $f(x) = \log_3(x)$, $g(x) = \log_{20}(x)$.
(e) $f(x) = 2^{\log_3 x}$, $g(x) = 3^{\log_{20} x}$.

Solution : (a) None. When x keeps increasing, $(\cos(x))^2$ periodically reaches zero. When $(\cos(x))^2$ is zero, $\sin(x)$ could be -1 or 1. So, no constant c that makes $f(x) \leq c \cdot g(x)$ or $f(x) \geq c \cdot g(x)$.

(b) (ii), $f(x) = O(g(x))$. We have $f(x) \leq 4g(x)$.

(c) (iii), $f(x) = \Omega(g(x))$. $f(x) \geq 0.5g(x)$.

(d) (i)(ii)(iii), $f(x) = \Theta(g(x))$. $\log_{20}(x) = \log_3(x)/\log_3 20$ where $1/\log_3 20$ is a constant.

(e) (iii), $f(x) = \Omega(g(x))$. $f(x) = 2^{\log_3 x} = x^{\log_3 2}$. $g(x) = 3^{\log_{20} x} = x^{\log_{20} 3}$. The answer is then obvious.

Grading Scheme: 3 for each.

Problem 9: [8 pts] Given two binary numbers a and b, $a \wedge b$ is the bitwise AND operation of a and b. E.g. if $a = 3_{10} = 011_2$ and $b = 6_{10} = 110_2$, then $a \wedge b = 2_{10} = 010_2$.

Consider the following two algorithms, both of which count the number of 1's in the binary representation of n.

procedure $CountOneA(n$: binary number)

$a \leftarrow 0$

while $n \neq 0$

  if $n \bmod 2 = 1$ **then** $a \leftarrow a + 1$

  $n \leftarrow \lfloor n/2 \rfloor$

return a

procedure $CountOneB(n$: binary number)

$a \leftarrow 0$

while $n \neq 0$

  $a \leftarrow a + 1$

  $n \leftarrow n \wedge (n - 1)$

return a

For each algorithm's running time, select all that apply from the list below, and briefly justify your answer.

(1) $O(\log n)$; (2) $\Omega(\log n)$; (3) $\Theta(\log n)$; (4) $O(1)$; (5) $\Omega(1)$; (6) $\Theta(1)$.

Solution : Algorithm A's running time is proportional to the number of bits in the binary representation of n, so it's $\Theta(\log n)$, which is also $O(\log n)$, $\Omega(\log n)$, and $\Omega(1)$.

Algorithm B's running time is proportional to the number of 1's in the binary representation of n, which fluctuates between 1 and $\log n$, depending on the actual value of n. So it's $O(\log n)$ and $\Omega(1)$.

Grading Scheme: 4, 4

Bonus: [10 pts] Prove $|(0, 1)| = |(0, 1) \times (0, 1)|$.

Solution : We need to construct an injection from $(0, 1)$ to $(0, 1) \times (0, 1)$, as well as one the other way round. Then applying the Schröder-Bernstein theorem would complete the proof. An injection $f : (0, 1) \to (0, 1) \times (0, 1)$ is trivial, e.g., $f(x) = (x, 0)$. An injection $f : (0, 1) \times (0, 1) \to (0, 1)$ can be defined as follows. For any $x, y \in (0, 1)$, let $x = 0.x_1x_2 \cdots$ and $y = 0.y_1y_2 \cdots$ be their valid decimal representations (i.e., not ending with infinitely many 9's). Define $f(x, y) = 0.x_1y_1x_2y_2 \cdots$. Note that this must also be a valid decimal representation of some real number in $(0, 1)$, because if it's not, $x = 0.x_1x_2\cdots$ and $y = 0.y_1y_2\cdots$ must end with infinitely many 9's. For any $(x, y) \neq (x', y')$, they must have at least one different digit, so $f(x, y) \neq f(x', y')$, hence f is an injection.