Question 1 Set Theory

(a)   i. <u>Describe the following set using the listing method:</u>
$$A = \{x : x \in Z \text{ and } 0 \le x^3 < 100\}$$
A = {0, 1, 2, 3, 4};

       ii. <u>Rewrite the following set using the set builder method:</u>
$$B = \{-1, 1/2, -1/3, 1/4, -1/5, 1/6, …\}$$
B = $\{\frac{(-1)^x}{x} : x \in Z^+\}$.

(b)   In a survey of 200 student, it was found that: 150 students took programming (P), 80 students took mathematics (M), 55 students took art (A), 60 students took mathematics and programming (M & P), 25 students took art and mathematics (A & M), 40 students took art and programming (A & P), and 15 students took art, mathematics and programming (A & M & P).

    i. <u>Draw a Venn diagram to display this information.</u>
Figure 1 - A Venn diagram for this exercise (not up to scale);
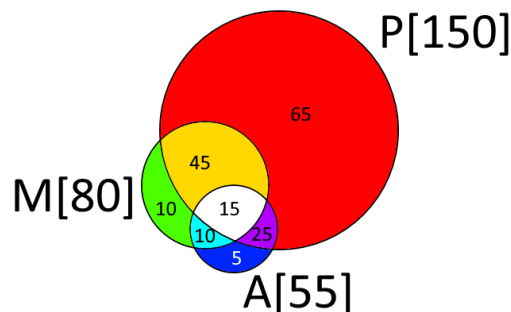
We can find the number of students that took only P and M (Yellow) as P∩M - P∩M∩A = 60 – 15 = 45;
The number of students that took only P and A (Violet) as P∩A - P∩M∩A = 40 – 15 = 25;
The number of students that took only M and A (Cyan) as M∩A - P∩M∩A = 25 – 15 = 10;

    ii. <u>Use Venn diagram to the find the number of students that took:</u>
1)Students that took **only Programming** (Red): P - P∩M - P∩A + P∩M∩A = 150 – 60 – 40 + 15 = **65**;
2)Students that took **two modules only**: P∩M + P∩A + M∩A – 3*P∩M∩A = 60 + 40 + 25 – 3*15 = **80**;
Or alternatively (Yellow + Violet + Cyan): 45 + 25 + 10 = 80;
3)Students that took **mathematics and programming but not art**: P∩M – P∩M∩A = 60 – 15 = **45**;
Or alternatively (Yellow): 45;

(c) <u>Let A and B be two subsets of the universal set U. Prove or disprove that: P(A ∪ B) = P(A) ∪ P(B).</u>
Let us provide a counter example to disprove that P(A ∪ B) = P(A) ∪ P(B).
Let A = {a, b} and B = {x}. Then the powersets of this sets are: P(A) = {∅,{a},{b},{a,b}}, P(B) = {∅,{x}}.
The conjunction A ∪ B = {a, b, x} and P(A ∪ B) = {∅, {a}, {b}, {x}, {a,b}, {a,x}, {b,x}, {a,b,x}}.
On the other hand, P(A) ∪ P(B) = { ∅, {a}, {b}, {x}, {a,b}}. **Therefore P(A ∪ B) ≠ P(A) ∪ P(B)**.
We can also disprove using the fact that the cardinality of the powerset of the set S is equal to the 2^n, where n is the number of the elements in the S. So, |P(A)| = 2^|A|, |P(B)| = 2^|B| and | P(A) ∪ P(B) | = 2^|A| + 2^|B| - 1(for ∅) – n (n ∈ A and n ∈ B). |P(A ∪ B)|= 2^|A ∪ B|. With this in mind let us try to disprove using cardinality: If LHS and RHS of the equation are the same, then they both has same elements and length and their cardinality should be equal => | P(A) ∪ P(B) | = P(A ∪ B)|. Solving it:
2^|A| + 2^|B| - 1 – n = 2^|A ∪ B|. Let |A| = x and |B| = y, |A ∪ B|= x + y. Then:
2^x + 2^y – 1 – n = 2^(x + y) => 2^x + 2^y - 2^x * 2^y = 1 + n => Because x, y and n are natural numbers we see that RHS are < 1 (if A and B are not empty sets) while LHS is ≥ 1, so this equation does not holds, **therefore P(A ∪ B) ≠ P(A) ∪ P(B)**.

(d) <u>Let A and B be two subset of a universal set U. Show that: A ⊆ B ⇔ A'∪B = U.</u>

From the definition of the complement, we can define that U = A ∪ A' (1) and U = B ∪ B' (2). From the definition of the subset: if $A \subseteq B \Rightarrow B = A \cup (B - A)$ (3).

    1) First, we prove that if A ⊆ B then A' ∪ B = U. If we know that A is the subset of B, then: (3) -> A' ∪ B = U => A' ∪ [A ∪ (B – A)]. Using associative property of the set we can write it like [A' ∪ A] ∪ (B – A). Now we use (1): U ∪ (B – A) or simply U if we use the property of the universal set.

    2) Now, we try to prove that if A' ∪ B = U then A ⊆ B. Start with (1): A' ∪ B = U => A' ∪ B = A ∪ A' => A = B => A ⊆ B

    Therefore, we proved **A ⊆ B ⇔ A'∪B = U**.

Question 2 Functions

(a)   Let f: R -> Z where $f(x) = \lfloor x/2 \rfloor$

    i. <u>Find f(1):</u>

    f(1) = $\left\lfloor \frac{1}{2} \right\rfloor = \lfloor 0,5 \rfloor = $ **0**;

    ii. <u>What is the set of pre-images of 10:</u>

    S = {x: x ∈ Z and 20 ≤ x < 22}, or **S = {20, 21}**;

    iii. <u>Say whether or not f(x) is injective(one-to-one), justifying your answer:</u>

    The f(x) is not an injective function because by the definition function is called injective if and only if $f(x_1) = f(x_2)$ implies that $x_1 = x_2$. In other words: a unique element from the domain map to a distinct element in co-domain, therefore there should be only one pre-image for every element in the image set. But we already showed early that that there are 2 pre-images for a single value "10" (etc. f(20) = f(21) = 10) => **f(x) is not injective**.

    iv. <u>Say whether or not f(x) is surjective(onto), justifying your answer:</u>

The f(x) is a surjective function because by the definition function is called surjective if for every element in co-domain there is a pre-image in the domain. We know that co-domain of f(x) (integers) is the subset of the set of all rational numbers (the domain of f(x)). Also, we know that every rational number can be expressed as an irreducible fraction of integers 'a' and 'b' (b≠0). So, based on that knowledge we can say that for every integer (an element of the co-domain) there exist another integer 2 time bigger (a = 2n), such that $\left\lfloor \frac{a}{2} \right\rfloor = \left\lfloor \frac{2n}{2} \right\rfloor = \lfloor n \rfloor = n \in Z$. Every element in the co-domain has a pre-image, therefore **f(x) is surjective function**.

(b)   Given a function g: R -> R is defined by g(x) = 3x + 5

    i. <u>Show that the function g is a bijection:</u>

    To show that a function is a bijection we must show that it is injective and surjective at the same time. To prove that the function is injective we need to show that if $g(x_1) = g(x_2)$ then $x_1 = x_2$. Let us try to test this: We assume that $g(x_1) = g(x_2)$ => $3x_1 + 5 = 3x_2 + 5$ => $3x_1 = 3x_2$ => There are no two distinct values x_1 and x_2 in the domain $D_g = R$ that will suffice this equation => $x_1 = x_2$ => g(x) is injective.

To show that g(x) is surjective we need to show that there are no values in co-domain that does not have a pre-image. g(x) = 3x + 5 => $x = \frac{g(x) - 5}{3}$, hence for every real number in co-domain of g(x) there is exist unique $x = \frac{g(x) - 5}{3}$ that is in D_g, and we can conclude that g(x) is surjective. **Because g(x) both injective and surjective it is a bijection.** We also can use the fact that g(x) is a linear function with domain R, so it must be bijective.

ii. <u>Find g^{-1}</u>

The function is invertible if and only if it is injective. We proved that g(x) is a bijective, so it is also invertible. g(x) = 3x + 5 => x = 3g^{-1}(x) + 5 => **g^{-1}(x) = $\frac{x-5}{3}$**, g^{-1}(x): R -> R.

(c)   Let f: D$_f$ -> [0; +∞) be a bijective function with f(x) = ln (x+1)

i. <u>Find the domain D$_f$ of this function:</u>

The logarithmic function is defined if an argument is > 0 => (x+1) > 0 => x > -1. But we also know that the range is described as [0; +∞), and that logarithmic functions are monotone increasing functions, so to find the lower bound of the domain we solve the equation: f(x) = 0 => ln(x+1) = 0 => x+1 = e^0 => x+1 = 1 => x = 0; **D$_f$** = {x: x > -1} ∩ {x: x ≥ 0} = {x: x ≥ 0} = **[0; +∞)** and f(x): [0; +∞) -> [0; +∞);

ii. <u>Find the inverse function f^{-1}:</u>

To find the inverse we first need to prove that f(x) is injective function => if f(x$_1$) = f(x$_2$) implies x$_1$ = x$_2$ then we can find the inverse. Test it: ln (x$_1$ + 1) = ln (x$_2$ + 1) => ln(x$_1$ + 1) − ln(x$_2$ + 1) = 0 => ln($\frac{x_1+1}{x_2+2}$) = 0 => $\frac{x_1+1}{x_2+2}$ = e^0 = 1 => x$_1$ + 1 = x$_2$ + 1 => x$_1$ = x$_2$ => f(x) is injective. Then, to find the inverse: f(x) = ln (x + 1) => x = ln (f^{-1}(x) + 1) => e^x = f^{-1}(x) + 1 => **f^{-1}(x) = e^x − 1**, f^{-1}(x): [0; +∞) -> [0; +∞).

iii. <u>Plot the curves of both function, f and f^{-1} in the same graph:</u>

Figure 2 – f(x) and f^{-1}(x)
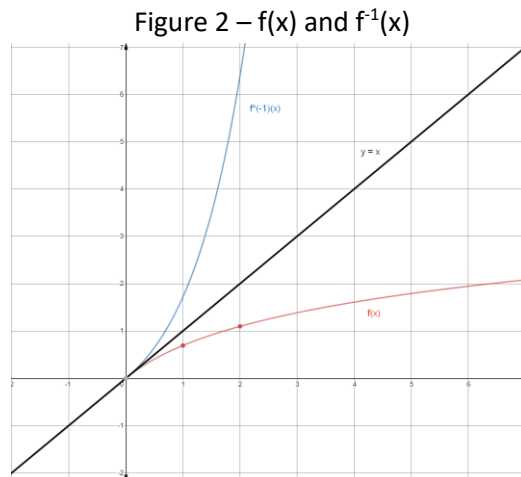
iv. <u>What can you say about these two curves?</u>

The functions **f(x) and f^{-1}(x) are reflected over the line y = x**.

(d)   Determine whether each of the following functions, defined from Z x Z to Z, is one-to-one, onto or both. Explain your answers.

i. <u>f(x, y) = x^2 + 1</u>

First, we check this function to be injective. To prove that the function f(x, y) is not injective we could show that there are two different pairs of elements from the domain that have the same image. Let us check pairs (1, 1) and (-1, 100): f(1, 1) = 1^2 + 1 = 2, f(-1, 100) = (-1)^2 + 1 = 2 => f(1, 1) = f(-1, 100) => f(x, y) is not injective. To prove that the function f(x, y) is not surjective we can find an element in the co-domain that has no pre-images from the domain. f(x, y) = x^2 + 1, and for every x ∈ Z the value of x^2 ≥ 0, so f(x, y) ≥ 0 and therefore any negative number from the co-domain Z has no pre-images, so we conclude that **f(x, y) is not injective nor surjective**.

ii. <u>g(x, y) = x + y + 2</u>

Similarly we provide a counterexample to prove that g(x, y) is not injective. The pairs (-1, -1) and (0, -2) are both in the domain Z x Z and distinct. g(-1, -1) = (-1) + (-1) + 2 = 0, g(0, -2) = 0 + (-2) + 2 = 0, so g(-1, -1) = g(0, -2) = 0 => g(x, y) is not injective. Now we prove that g(x, y) is surjective: g(x, y) = x + y + 2 => x + y = g(x, y) – 2. LHS of this equation is a sum of two integers that is integer itself, and RHS is also an integer. We can find any number of pairs of integers x and y that the sum will be equal to another integer (g(x) – 2). Therefore, **g(x, y) is not injective but surjective**.

Question 3 Propositional Logic

(a)   Let p, q, r and s four propositions. Assuming that p and r are false and that q and s are true, find the truth value of each of the following propositions.
        i. ((p ∧ ¬q) → (q ∧ r)) → (s ∨ ¬q)
        Assuming, that p = r = F and q = s = T we can rewrite ((p ∧ ¬q) → (q ∧ r)) → (s ∨ ¬q):
((F ∧ ¬T) → (T ∧ F)) → (T ∨ ¬T) ≡ ((F ∧ F) → F) → (T ∨ F) ≡ (F → F) → T ≡ T → T ≡ T. Therefore, **the value of this proposition is "True"**.

        ii. ((p ∨ q) ∧ (q ∨ s)) → ((¬r ∨ p) ∧ (q ∨ s))
        Assuming, that p = r = F and q = s = T we can rewrite ((p ∨ q) ∧ (q ∨ s)) → ((¬r ∨ p) ∧ (q ∨ s)):
((F ∨ T) ∧ (T ∨ T)) → ((¬F ∨ F) ∧ (T ∨ T)) ≡ (T ∧ T) → (T ∧ T) ≡ T ∧ T ≡ T. Therefore, **the value of this proposition is "True"**.

(b)   Let p and q be two propositions defined as follows: p means 'A student can take the algorithm module' whereas q means 'Student passes discrete mathematics'. Express each of the three following compound propositions symbolically by using p, q and appropriate logical symbols.

        i. 'A sufficient condition for a student to take the algorithm module is that
they pass discrete mathematics' = **q -> p**.

        ii. 'A student can take the algorithm module only if they pass discrete
mathematics' = **p -> q**.

        iii. 'A student can take the algorithm module if they pass discrete mathematics' = **q -> p**.

        iv. 'A student either passes discrete mathematics or can take the algorithm
module' = **q ∨ p** (or q ⊕ p if this sentence form assume that a student can take only one option).

(c)   Write in words and express symbolically in terms p and q, defined in (a), the contrapositive, the converse and the inverse of the implication: 'A student can take the algorithm module if they pass discrete mathematics'.
        The converse: 'If a student can take the algorithm module, he is passed discrete mathematics', expressed symbolically as **[p -> q]**.

        The inverse: 'If a student not passed discrete mathematics, he can't take the algorithm module', expressed symbolically as **[¬q -> ¬p]**.

        The contrapositive: 'If a student can't take the algorithm module, he is not passed discrete mathematics', expressed symbolically as **[¬p -> ¬q]**.

(d)   Consider the following three propositions: S means "Samir goes to the party ", C means "Callum goes to the party" and J means "Jay goes to the party". Express each of the three following compound propositions symbolically by using c, j, s and appropriate logical symbols.
        i. "Samir goes to the party only if both Callum and 'Jay aren't going to
the party.", expressed symbolically as: **S -> (¬C ∧ ¬J)**.

ii. "Either both Samir and Jay go to the party or Callum goes to the party, but not both", expressed symbolically as: **(S ∧ J) ⊕ C**.

(e)   A tautology is a proposition that is always true. Let p and q be two propositions, show that (p → q) ⇔ (¬q → ¬p) is a tautology:

   To show that (p → q) ⇔ (¬q → ¬p) is a tautology, we need to prove that this statement is holds true for all p and q. We prove this using truth table, but alternatively we can use the fact that (¬q → ¬p) is contrapositive of the statement (p → q) and (¬q → ¬p) ≡ (p → q). Therefore, we can rewrite initial statement as (p → q) ⇔ (p → q). We can say that this statement is a tautology **because both sides are either True or False, and biconditional statement will always be true**.

Table 1 – Truth table for (p → q) ⇔ (¬q → ¬p)

p	q	p -> q	¬q	¬p	¬q → ¬p	(p → q) ⇔ (¬q → ¬p)
1	1	1	0	0	1	1
1	0	0	1	0	0	1
0	1	1	0	1	1	1
0	0	1	1	1	1	1

Question 4 Predicate logic

(a)   Let P(x, y) be a Boolean function. Assume that ∀x∃yP(x, y) is True and that the domain of discourse is nonempty. Which of the following must also be true? If the statement is true, explain; otherwise, give a counterexample.

   i. ∀x∀yP(x, y):

   The statement ∀x∀yP(x, y) can be true (if P(x, y) is a tautology), depending on a function P(x, y), but it is not given from the initial assumption that ∀x∃yP(x, y) = True. To disprove it, we can offer a counter example: let P(x, y) = x + y. For this function ∀x∃yP(x, y) = True, but ∀x∀yP(x, y) = False, therefore, **the initial statement is not necessary true from the premise**.

   ii. ∃x∀yP(x, y):

   The statement ∃x∀yP(x, y) is also can be true (for example previous P(x, y) = x + y), but not given. Again, let us provide counter example: let P(x, y) = x'y + xy'. For this function ∀x∃yP(x, y) = True, but ∃x∀yP(x, y) = False, therefore, **the initial statement is not necessary true from the premise**.

   iii. ∃x∃yP(x, y):

   The statement **∃x∃yP(x, y) must be true** for any P(x, y) that satisfy initial implication ∀x∃yP(x, y) = True, because ∃x∃yP(x, y) is a special case of ∀x∃yP(x, y). So, if this statement is true for all x, then it is true for at least one.

Table 2 - Truth table for P(x, y)

x	y	x + y	x'y + xy'
1	1	1	0
1	0	1	1
0	1	1	1
0	0	0	0

(b)   Given the following argument: "If it rains then the concert will be cancelled", "The concert was cancelled, therefore it rained", Assume p means " it rains" whereas q means "concert cancelled".

   i. Translate this argument to a symbolic form:
   Premise: **p -> q**;
   Conclusion: **q -> p**.

   ii. Construct the truth table:

Table 3 - Truth table for p -> q and q -> p

p	q	p -> q	q -> p
1	1	1	1
1	0	0	1
0	1	1	0
0	0	1	1

iii. <u>Determine if this argument is a valid argument or not:</u>
This argument is not valid. This sort of fallacy is called affirming the consequent. The statement q -> p is a converse of p -> q. From the first statement we can derive that p is sufficient for q, but not necessary. From the truth table of the statement p -> q we can see that even when p is not true the q can be true or false, and the implication will hold. For example, concert can be cancelled for another reason, so if we only know that there was no concert, we can't say if it rained with certainty.

(c)   <u>Let p, q, r, s and t be statements variables. Use the valid argument forms</u>
<u>to deduce the conclusion, ¬q, from the premises, giving a reason for each</u>
<u>step.</u>

(a) ¬p ∨ q → r
(b) s ∨ ¬q
(c) ¬t
(d) p → t
(e) ¬p ∧ r → ¬s
——————————————————
(f) ∴ ¬q

1) We use modus tollens on (d) p -> t and (c) ¬t to reason out: ¬p;
2) We use modus ponens on (a) (¬p ∨ q) -> r and (1) ¬p to reason out: r;
3) We use conjunctive syllogism on (1) ¬p and (2) r to reason out: (¬p ∧ r);
4) We use modus ponens on (e) (¬p ∧ r) -> ¬s and (3) (¬p ∧ r) to reason out: ¬s;
5) We use disjunctive syllogism on (b) s ∨ ¬q and (4) ¬s to reason out: ¬q.

Question 5 Boolean Algebra
(a)   <u>What is the value of the Boolean expression (x + y)(x'.y')?</u>
Using distributive law we can write (x + y)(x'.y') as (x.(x'.y'))+(y.(x'.y')). Next, we apply associativity law: (x.(x'.y'))+(y.(x'.y')) = ((x.x').y')+(x'.(y.y')). From complementation law we can derive x.x' = y.y' = 0, so ((x.x').y')+(x'.(y.y')) = (0.y') + (x'.0). We then simplify it using annihilation law: (0.y') + (x'.0) = 0 + 0 and conclude 0 + 0 = 0 using idempotent law, proving that **the value of the Boolean expression is 0 or False**.

(b)   Consider the following combinatorial circuit with three inputs x, y and z, and one output f(x, y, z)
         i. <u>Write the output f(x, y, z) in its disjunctive normal form:</u>
         Figure 3 – Combinatorial circuit: **f(x, y, z) = (xyz + xyz') + xy'z'**
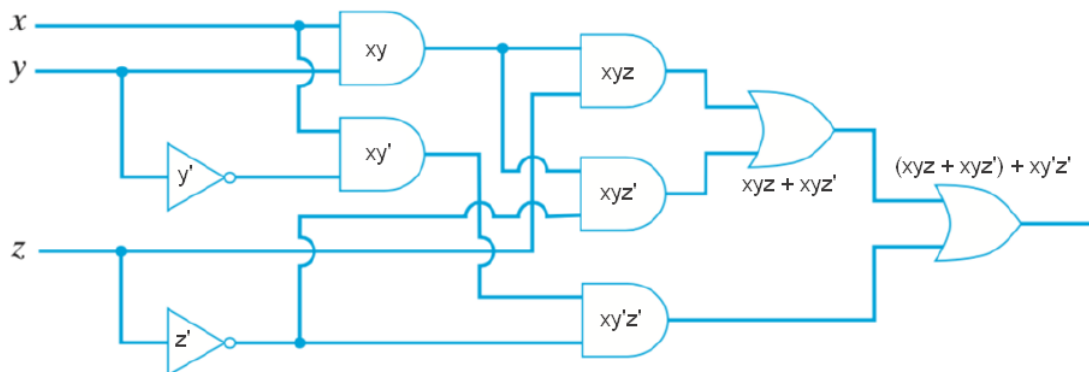
ii. Fill in the missing output value in the following table:
Table 4 -Truth table for f(x, y, z) table:

x	y	z	f(x, y, z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

iii. Show that f(x, y, z) can be simplified to f(x, y, z) = x(y + z'):
f(x, y, z) = (xyz + xyz') + xy'z' = (xy.(z + z')) + xy'z' [distributive law] = xy.1 + xy'z' [complement law] = xy + xy'z' [identity law] = x(y + (y'z')) [distributive law] = x((y+y')(y+z')) [distributive law] = x(1.(y+z')) [complementation law] = **x(y+z')** [identity law].

iv. Draw the simplified circuit equivalent to f(x, y, z):
Figure 4 – Simplified combinatorial circuit: f(x, y, z) = x(y+z')
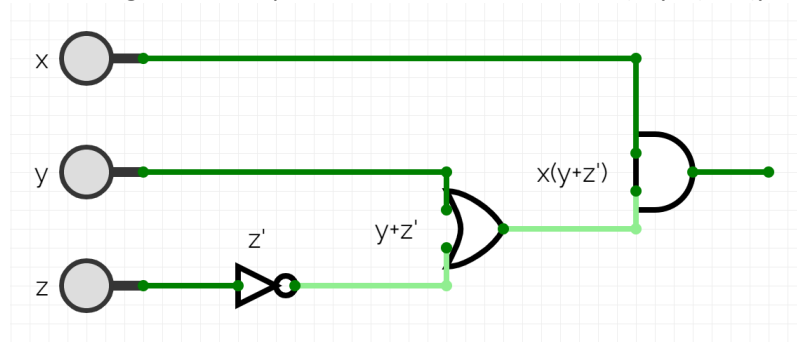
(c)   i. What is the advantage of using Karnaugh map (K-map)?
The Karnaugh map provides an **easy way to simplify** the Boolean expression to its disjunctive normal form (DNF). It allows to reduce need for extensive calculations and save the time.

ii. Fill in the following K-map for the Boolean function:
Table 5 - The Karnaugh map for the function F(x, y, z) = x'y'z + x'yz' + xyz' + xy'z'

z \ xy	00	01	11	10
0		1	1	1
1	1			

iii. Use the previous K-map and find a minimization, as the sum of three terms, of the expression
There are two ways to minimize the initial expression as sum of the three terms: F(x, y, z) = x'y'z + x'yz' + xyz' + xy'z'
1) f(x, y, z) = x'y'z + (x'yz' + xyz') + xy'z' = x'y'z + (x' + x)yz' + xy'z' [distributive law] = x'y'z + 1.yz' + xy'z' [complement law] = **x'y'z + yz' + xy'z'** [identity law];
2) f(x, y, z) = x'y'z + x'yz' + (xyz' + xy'z') = x'y'z + x'yz' + x(y + y')z' [distributive law] = x'y'z + x'yz' + x.1.z' [complement law] = **x'y'z + x'yz' + xz'** [identity law].