# UoL logoUniversity of London

# Assessment Coversheet

Complete this coversheet and read the instructions below carefully.

**Candidate Number**:

WP1282

**Degree Title**:

BSc Computer Science

**Course/Module Title**:

Discrete Mathematics

**Course/Module Code:**

CM1020

**Enter the numbers, and sub-sections, of the questions in the order in which you have attempted them:**

**Question 1**: a, b, c [i, ii, iii, iv], d [I, ii, iii, iv], e [i, ii], f, g

**Question 2:** a [i, ii, iii, iv], b[i, ii], c[i,ii]

**Date**: 08.03.2021

**Instructions to Candidates**

1. Complete this coversheet and begin typing your answers on the page below, or, submit the coversheet with your handwritten answers (where handwritten answers are permitted or required as part of your online timed assessment).
2. Clearly state the question number, and any sub-sections, at the beginning of each answer and also note them in the space provided above.
3. For typed answers, use a plain font such as Arial or Calibri and font size 11 or larger.
4. Where permission has been given in advance, handwritten answers (including diagrams or mathematical formulae) must be done on light coloured paper using blue or black ink.
5. Reference your diagrams in your typed answers. Label diagrams clearly.

**The Examiners will attach great importance to legibility, accuracy and clarity of expression.**

**Begin your answers on this page**

**QUESTION 1**

1. Let start with small explanation: can be translated as “all elements in A plus all elements that are in B minus all elements that are both in A and B” (inclusion-exclusion principal). We can rewrite it using set notations as:

From this we can find:

So, .

1. Because we need to describe the area that comprise of many elements it is easy to do it using subtraction. is related to the set representing the area comprising the regions 001, 010, 100, 011, 110, 101, 111. To find required area we just exclude set representing areas 110 and 111. One way to describe this is: .
2. We use the fact that p = T, q = T and r = F. We also will use the following facts about logical operations:

(\*)Implication(p→q) is only false when antecedent(p) is True and consequent(q) is False.

(\*\*)Logical biconditional (p↔q) is true when both statements (p and q) have same truth values (both true or both false).

(\*\*\*)Exclusive OR (pꓴq) on the contrary is false when both statements (p and q) have same truth values.

(\*\*\*\*)Conjunction (pꓵq) is true only when both statements (p and q) are true.

1. ;
2. ;
3. **;**
4. **;**
5. We will use following arguments: x1 = 2 and x2 = 3 are both elements of the universe of discourse (x1, x2 ϵ R). Let us also write down truth values of statements:

(\*) P(x1): 22 > 1 ≡ 4 > 1 ≡ True, Q(x1): 2 + 1 < 4 ≡ 3 < 4 ≡ True;

(\*\*) P(x2): 32 > 1 ≡ 9 > 1 ≡ True, Q(x2): 3 + 1 < 4 ≡ 4 < 4 ≡ False.

1. . To prove universal quantifier () wrong we need to prove it wrong for one x. We test it using x2 = 3. From (\*\*) we can write (P(x)→Q(x)) as (P(x2)→Q(x2)) ≡ (T→F) ≡ False. So,  **is FALSE**.
2. To prove existential quantifier () true, we need to show it true for at least one x. We test it using x1 = 2. From (\*) we can write (P(x)→Q(x)) as (P(x1)→Q(x1)) ≡ (T→T) ≡ True. So,  **is TRUE**.
3. Again, one counter example is enough to prove universal quantifier to be false, and again we use x2 = 3. From (\*\*) we can write (P(x2)ꓵQ(x2)) ≡ (TꓵF) ≡ False. So,  **is FALSE**.
4. “There exists at least one x, such that P(x) is true and Q(x) is false. To prove existential quantifier, we again provide one example that make this statement true and again we use x2 = 3. From (\*\*) we can write (P(x2)ꓵ¬Q(x2))≡(Tꓵ¬F) ≡ (TꓵT) ≡ True. So,  **is True**.
5. i. Premise 1: “If it snows, then school is closed (p→q)”. Premise 2: “School is open (¬q)”. Conclusion: “It is not snowing (¬p)”. This is **VALID** argument based on the **law of contraposition** (p→q ≡ ¬q → ¬p).

ii. Premise 1: “If the movie is long, I will fall asleep (p→q)”. Premise 2: “I do fall asleep (q)”. Conclusion: “The movie was long (p)”. This is **INVALID** argument based on the fallacy known as “**Converse error**” (p→q ≠ q → p). For example: “Movie was actually short, but I fell asleep because I was tired.”

1. There are several ways to prove this equivalence. To make this proof a little easier to read let us switch (AꓵB) for D. Using De Morgan’s law, we can also show that . Using this information, we can write the initial expression as , and switch variables to get: . Now, using distributive law: After that, using negation law:.Applying identity law:. We already showed that application of the De Morgan’s law , so we can switch back:. Because disjunction is associative, we can rewrite this as: (. And with this we proved equivalence as required.
2. Because this task can be done in A ways or B ways or … we can use addition principle and treat the total number of ways this task can be done as:

Ptotal = P1question+ P2questions+ P3questions+ P4questions+ P5questions+ P6questions.

Because order in which questions are done is important and we assume that there no repetitions we can find the number of ways to do subtasks as permutations of k elements from n elements using formula P(n,k) = n!/(n-k)!

P1question = P(6,1) = 6!/(6-1)! = 6!/5! = 6\*5!/5! = 6;

P2questions = P(6,2) = 6!/(6-2)! = 6!/4! = 6\*5\*4!/4! = 30;

P3question = P(6,3) = 6!/(6-3)! = 6!/3! = 6\*5\*4\*3!/3! = 120;

P4questions = P(6,4) = 6!/(6-4)! = 6!/2! = 6\*5\*4\*3\*2!/2! = 360;

P5question = P(6,5) = 6!/(6-5)! = 6!/1! = 6\*5\*4\*3\*2\*1/1 = 720;

P6questions = P(6,6) = 6!/(6-6)! = 6!/0! = 6\*5\*4\*3\*2\*1/1 = 720; (We treat the way to answer 5 questions 1,2,3,4,5 and 6 questions 1,2,3,4,5,6 as different ways)

So, the total number of ways **Ptotal = 6+30+120+360+720+720 = 1956** ways to attempt one or multiple questions in the exam.

**QUESTION 2**

1. i. The gates of the logical circuit are enumerated as on the figure 1.

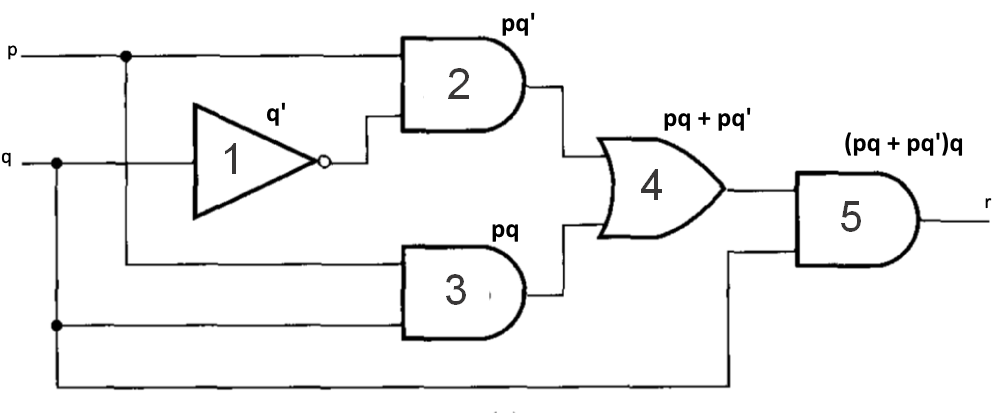


Figure 1 – Initial logical circuit and logical gates

**Gate 1 - NOT gate.**

**Gates 2, 3, 5 – AND gates.**

**Gate 4 – OR gate.**

ii. All inner level expressions can be seen on the figure 1. And the final expression of the output **r = (pq + pq’)q**.

iii. r = (pq + pq’)q = (p(q+q’))q [Using distributive law] = (p.1)q [Negation law] = (p)q [Identity law] => **r = pq** [associativity property of the Boolean product]

iv. Simplified logical circuit is shown on the figure 2.

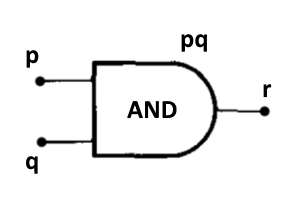


Figure 2 – Simplified logical circuit

1. i. Bijection is a function that pairs elements of two sets in such way that each element of the one set is paired exactly with one element of the other set and there no unpaired elements in both sets. **For function to be considered bijection it must be injective and surjective** at the same time.

Function f(x) is injective or one-to-one function if preimage of any element of the codomain is unique element from the domain.

Function f(x) is surjective or onto function if every element in the codomain has at least one preimage in the domain of the function.

ii. To show that f(x) = 2x+3 is a bijection we need to prove that it is injective and surjective. To prove that f(x) is injective we show that the only way that f(a) = f(b) is that a = b. Let show that If f(a) = 2a+3 and f(b) = 2b+3 then a = b. 2a+3 = 2b+3 => a + 3 = b + 3 => a = b => There no 2 unique values in the domain that have same image => **f(x) is injective**.

To show that f(x) is surjective we need to prove that every y in the codomain has at least one preimage in the domain. y = 2x+3 => log2y = x + 3 => x = log2y – 3. For all y ϵ R+, log2y is defined and log2y – 3 will result in a real number (domain of f(x)), so we conclude that **f(x) is surjective**.

And because f(x) is injective and bijective we proved that it is **bijective**.

The inverse of the function f(x) is a function f-1(x) that maps all elements from the codomain to the elements of the domain, such that f-1(y) = x. Not all functions have an inverse, but we already proved that f(x) is bijective, so it has an inverse, and we can write it down as **f-1(x) = log2x – 3**.

1. for all n ϵ Z+.
2. **S2** = 1\*1! + 2\*2! = 1 + 4 = **5**.

**S3** = 1\*1! + 2\*2! + 3\*3! = 1 + 4 + 18 = **23**.

1. Prove that Sn = (n+1)! – 1 for all n ϵ Z+. Proof by induction include 3 major steps:

* Base case: S1 = 1\*1! = 1 from definition and from the assumption S1 = (1 + 1)! – 1 = 2! – 1 = 2 – 1 = 1. As we see, **base case is true**.
* Inductive hypothesis: Assume that for n = a, **Sa = (a+1)! – 1** is **true** (\*).
* Inductive step: If we prove that Sa+1 = ([a+1] +1)! – 1 (\*\*) based on our hypothesis, we can prove Sn = (n+1)! – 1 by induction.

From the definition we can describe Sn+1 as:

Using the inductive hypothesis (\*) we can rewrite this as:

Now we return to the inductive step (\*\*):

(\*\*\*\*)

So, to complete the inductive step we prove that (\*\*\*) is equal to the (\*\*\*\*):

We can rewrite this as:

And now, we can substitute [a+2] for m to see:

This form is just the definition of the factorial and we know that it is true =>

**We proved inductive step Sa+1 based on the hypothesis that Sa is true, so by the principal of the mathematical induction Sn = (n+1)! – 1 for all n ϵ Z+**.