Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

BACHELOR OF SCIENCE (HONS) IN
- APPLIED COMPUTING
- COMPUTER FORENSICS & SECURITY
- ENTERTAINMENT SYSTEMS
- THE INTERNET OF THINGS

EXAMINATION:

DISCRETE MATHEMATICS (COMMON MODULE)

SEMESTER 1 - YEAR 1

AUGUST 2025 DURATION: 2 HOURS

INTERNAL EXAMINERS:	**DR DENIS FLYNN**	**DATE:**	**25 AUG 2025**
	DR KIERAN MURPHY	**TIME:**	**11:45 AM**
		VENUE:	**MAIN HALL**
EXTERNAL EXAMINER:	**DR JULIE CROWLEY**		

INSTRUCTIONS TO CANDIDATES

1. **ANSWER ALL QUESTIONS.**
2. **TOTAL MARKS = 100.**
3. **EXAM PAPER (5 PAGES EXCLUDING THIS COVER PAGE) AND FORMULA SHEET (1 PAGE)**

MATERIALS REQUIRED

1. **NEW MATHEMATICS TABLES.**
2. **GRAPH PAPER**

SOUTH EAST TECHNOLOGICAL UNIVERSITY

Question 1

(a) Convert the following English sentence into a propositional logic expression with **atomic** propositions:

> *"If the alarm system is enabled and either the camera recognises an intruder or a window is broken, then the Gardaí will be notified."*

(4 marks)

(b) Which of the following are well-formed propositional formulas?
(i) $\wedge\neg pq$ **(ii)** $(p \vee q) \rightarrow \neg r$ **(iii)** $\neg(p \rightarrow)q$ **(iv)** $p \wedge (q\vee \rightarrow s)$
Justify your answers.

(4 marks)

(c) Evaluate each of the following
(i) $\sum_{k=0}^{4}(2k+3)$ **(ii)** $\sum_{k=2}^{5} 3^{k-1}$ **(iii)** $\prod_{k=1}^{4} \frac{k}{k+1}$

(6 marks)

(d) How many shortest lattice paths start at $(4, 2)$ and

(i) end at $(9, 8)$?

(ii) end at $(9, 8)$ and pass through $(6, 5)$?

(iii) end at $(9, 8)$ and avoid $(6, 5)$?

([3×2] 6 marks)

Question 2

(a) Consider the functions defined by the following Python code:
(recall that // is integer division)

```
def f(x):
  return (x + 2) ** 2 - 9
def g(x):
  return 3 * x - 1
def h(x):
  return x // 3
def j(x):
  return x % 5
def k(x):
  return 4 - x
```

Evaluate the following (note, show all work):

(i) $f(g(2))$ **(ii)** $h(k(8))$ **(iii)** $g(h(f(1)))$ **(iv)** $j(k(j(12)))$

(4 marks)

(b) Consider the sequence 8, 24, 40, 56, 72, 88, ...

(i) Construct a recursive definition for the sequence.

(ii) Construct a closed form definition the nth term of the sequence.

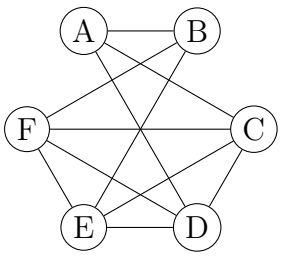
(iii) Is 2024 a term in the sequence?

(iv) How many terms of the sequence are less than 1000?

(v) Determine the sum of the first 100 terms of the sequence.

([5×2] **10 marks)**

(c) **(i)** For the following graph, determine its degree sequence.



(ii) Is the graph a simple graph? Justify your answer.

(iii) The **girth** of a graph is the length of its shortest cycle. Write down the girth of this graph.

(6 marks)

Question 3

(a) Consider the sets defined by the following Python code:

```
U = set(range(1, 12))
A = set(range(2, 10, 3))
B = set(range(1, 12, 2))
C = set(range(3, 8))

D = (A.union(B)).difference(C)
E = U.difference(A.intersection(B))
F = B.difference(A).union(C.difference(B))
```

(i) Write out the sets U, A, B, and C.

(ii) Use a Venn Diagram to represent the sets above.

(iii) Write down the equivalent mathematical expression for sets D, E, and F.

(iv) Compute the values for sets D, E, and F.

([$2 + 2 + 1 + 1$] 6 marks)

(b) Let $A = \{1, 2, 3, 4, 5, 6, 7\}$.

(i) How many subsets of A contain all the odd numbers?

(ii) How many subsets of A contain exactly two odd numbers?

(iii) How many subsets of A contain at least two odd numbers?

(iv) How many subsets of cardinality 5 contain exactly two even numbers?

([4×2] 8 marks)

(c) Are the statements $(P \to Q) \wedge (Q \to R)$ and $P \to (Q \wedge R)$ logically equivalent? Explain why.

(6 marks)

Question 4

(a) A network router uses the following rules to determine the quality of service (QoS) for different types of network traffic. Express each of these rules as a propositional logic expression, using the atomic propositions:

- V = "The traffic is video streaming."
- P = "The traffic has high priority."
- G = "The traffic is granted extra bandwidth."

(i) If the traffic is video streaming, then it has high priority.

(ii) If the traffic has high priority and is not granted extra bandwidth, then it is not video streaming.

(iii) If the traffic is granted extra bandwidth, then it is video streaming or it has high priority.

(iv) If the traffic is not video streaming and does not have high priority, then it is not granted extra bandwidth.

($[1 + 3 \times 2]$ 7 marks)

(b) The third term of a geometric progression is 2, and the sixth term is 16. Find the first term and the common ratio of this progression.

(3 marks)

(c) How many 12-bit strings (that is, bit strings of length 12) are there which satisfy each of the following criteria? Explain your answer.

(i) Start with the sub-string 110.

(ii) Have weight 5 (i.e. contain exactly five 1's in total) and start with the sub-string 110.

(iii) Have weight of 6 and are divisible by 8.

($[1 + 2 + 2]$ 5 marks)

(d) Use truth tables to show that the following proposition is a tautology:

$$(p \to (q \vee r)) \to ((p \wedge \neg q) \to r)$$

(5 marks)

Question 5

(a)

```
A = set(range(2,6))
R = {(a, b) for a in A for b in A if a % b == 0}
```

(i) From the Python code above, write out the set A and the relation R in set notation.

(ii) Represent the relation R as a digraph.

(iii) Is the relation R reflexive? symmetric? transitive? (Justify your answer).

(iv) Is R an equivalence relation? (Justify your answer).

(v) Determine whether the relation R is irreflexive? antisymmetric? asymmetric? (Justify your answer).

([3 + 2 + 3 + 2 + 3] 13 marks)

(b) What does the following function do? (Justify your answer).

```
def isWhat(A,B):
    for element in A:
        if element not in B:
            return(False)
    else:
        return(len(A)==len(B))
```

(4 marks)

(c) The first term of an arithmetic progression is 3, and the common difference is 5. What is the 12th term of this arithmetic progression?

(3 marks)

Laws of Logic

Logical Connective	Symbol	Python Operator	Precedence	Logic Gate
Negation (NOT)	$\neg$	**not**	Highest	
Conjunctive (AND)	$\wedge$	**and**	Medium	
Disjunctive (OR)	$\vee$	**or**	Lowest	

Basic Rules of Logic

Commutative Laws

$$p \vee q \Leftrightarrow q \vee p \qquad p \wedge q \Leftrightarrow q \wedge p$$

Associative Laws

$$(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r) \qquad (p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$$

Distributive Laws

$$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r) \qquad p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$$

Identity Laws

$$p \vee \mathbf{F} \Leftrightarrow p \qquad p \wedge \mathbf{T} \Leftrightarrow p$$

Negation Laws

$$p \wedge (\neg p) \Leftrightarrow \mathbf{F} \qquad p \vee (\neg p) \Leftrightarrow \mathbf{T}$$

Idempotent Laws

$$p \vee p \Leftrightarrow p \qquad p \wedge p \Leftrightarrow p$$

Null Laws

$$p \wedge \mathbf{F} \Leftrightarrow \mathbf{F} \qquad p \vee \mathbf{T} \Leftrightarrow \mathbf{T}$$

Absorption Laws

$$p \wedge (p \vee q) \Leftrightarrow p \qquad p \vee (p \wedge q) \Leftrightarrow p$$

DeMorgan's Laws

$$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q \qquad \neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$$

Involution Law

$$\neg(\neg p) \Leftrightarrow p$$

Implications and Equivalences

Detachment (Modus Ponens)

$$(p \rightarrow q) \wedge p \Rightarrow q$$

Indirect Reasoning (Modus Tollens)

$$(p \rightarrow q) \wedge \neg q \Rightarrow \neg p$$

Disjunctive Addition

$$p \Rightarrow (p \vee q)$$

Conjunctive Simplification

$$(p \wedge q) \Rightarrow p \qquad (p \wedge q) \Rightarrow q$$

Disjunctive Simplification

$$(p \vee q) \wedge \neg p \Rightarrow q \qquad (p \vee q) \wedge \neg q \Rightarrow p$$

Chain Rule

$$(p \rightarrow q) \wedge (q \rightarrow r) \Rightarrow (p \rightarrow r)$$

Resolution

$$(\neg p \vee r) \wedge (p \vee q) \Rightarrow (q \vee r)$$

Conditional Equivalence

$$p \rightarrow q \Leftrightarrow \neg p \vee q$$

Biconditional Equivalences

$$(p \leftrightarrow q) \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$$
$$\Leftrightarrow (p \wedge q) \vee (\neg q \wedge \neg q)$$

Contrapositive

$$p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$$

Python Cheat Sheet

Base Types

integer, float, boolean, string, bytes

int 163 0 −192 0b110 (binary) 0x3F (hex)

float 9.32 0.0 −1.7E−6 (×10⁻⁶)

bool False (0) True (1)

str 'some text' or "some text"

bytes b"text\xfe\775"

Container Types

ordered containers — repeatable values

list	[1,5,3]	["a",1,5,5]	[5]	[]
tuple	(1,5,3)	"a",1,5,5	(5,)	()

Immutable (non-modifiable values)

str	"153"	""

key containers — no order, unique keys

set	{"key1","key2"}	{1,9,3,0}	**set**()
dict	{"key1":value1,"key2":value2}	**dict**(a=3,b="v")	{}

Integer Sequences

range([start,] end [,step])

start default is 0 (inclusive), end (exclusive), step default is 1.

range(5) → 0,1,2,3,4
range(2,5) → 2,3,4
range(2,12,3) → 2,5,8,11
range(20,5,−5) → 20,15,10

Conversions

type(expression)

int('153') → 15
int('3f',16) → 63 (Specify base in 2nd parameter)
int(−11.24e8) → −1124000000
int(15.56) → 15 (Truncate decimal point)
round(15.58,1) → 15.6 (Round to 1 decimal place)
float('15.56') → 15.56

bool(x) (False for None, zero or empty containers)
str(x) (String representation of x.)

chr(65) → 'A' code ↔ char **ord**('A') → 65

list('abc') → ['a','b','c']
dict([(3,'three'), (1,'one')]) → {3:'three', 1:'one'}
set(['one','two']) → {'one','two'}

(Split string using a separator, **str** → **list** of **str**)
'random:data:666'.split(':') → ['random', 'data', '666']

(Join a list of strings, **list** of **str** → **str**)
':'.join(['random', 'data', '666']) → 'random:data:666'

(Convert each element in a collection)
[**int**(x) **for** x **in** ['1','29','−3']] → [1, 29, −3]

Operations on Sets

Operators

|	.union
&	.intersection
−	.difference

Methods

s.add(key)	s.update(s2)
s.clear()	s.remove(key)

Operations on Lists

Methods

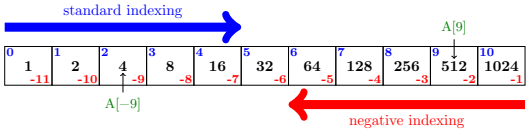
a.append(value)	a.extend(a2)
s.insert(idx,value)	a.pop()

Generic Operations on Containers

min(c) **max**(c) **sum**(c) **sorted**(c)
len(c) (Number of elements in collection c)

all(c) → True if **all** items in c evaluate to True, else False.
any(c) → True if **at least one** item in c evaluate to True, else False.

Sequence Containers Indexing



a[3:6] → [8, 16, 32]
a[1:−1] → [2, 4, 8, 16, 32, 64, 128, 256, 512]
a[::−1] → [1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1]

Looping over Collections

(Loop over values)
for value **in** A:
  print(value)

(Count and loop over values)
for k,value **in** **enumerate**(A):
  print(k, value)

(While loop)
k = 0
while k<**len**(A):
  print(k, A[k])
  k += 1

Initialisation **before** loop.
update **within** loop.

break immediatly exits loop. **continue** skips to next iteration.
else block for normal loop exit.