Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

## BACHELOR OF SCIENCE (HONS) IN
## - APPLIED COMPUTING
## - COMPUTER FORENSICS & SECURITY

EXAMINATION:

# DISCRETE MATHEMATICS (COMMON MODULE)

## SEMESTER 1 - YEAR 1

**DECEMBER 2023** **DURATION: 2 HOURS**

| | | | |
|---|---|---|---|
| INTERNAL EXAMINERS: | DR DENIS FLYNN<br>DR KIERAN MURPHY | DATE:<br>TIME:<br>VENUE: | 15 DEC 2023<br>11.45 AM<br>MAIN HALL |
| EXTERNAL EXAMINER: | MS MARGARET FINNEGAN | | |

INSTRUCTIONS TO CANDIDATES

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

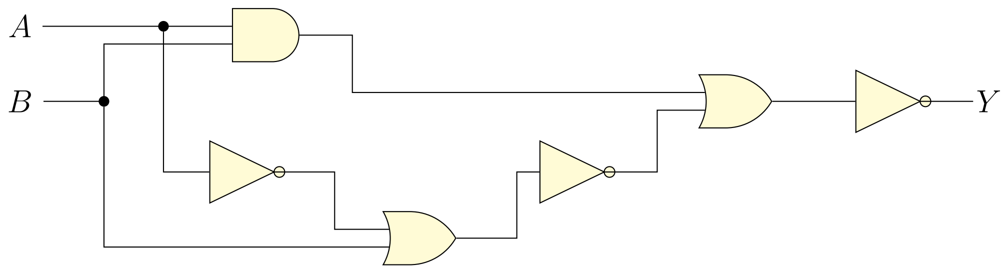
MATERIALS REQUIRED

1. NEW MATHEMATICS TABLES.
2. GRAPH PAPER

**SOUTH EAST TECHNOLOGICAL UNIVERSITY**

## Question 1

**(a)** Consider the following logical circuit with two inputs, $A$ and $B$, and output $Y$.



**(i)** Construct a logical expression to represent the output, $Y$.

**(ii)** Construct a truth table for this expression and classify the logical expression as a tautology, a contradiction, or satisfiable.

**(iii)** Compare the inputs to the output and construct a shorter (fewer logic gates) logical expression that is logically equivalent to the original circuit.

([$4+4+4$] **12 marks)**

**(b)** What does the following code output? (Justify your answer.)

```
A = {3, 5, 7}
B = {3, 5, 6, 7}

for x in A:
    if x not in B:
        print(False)
        break
else:
    print(len(A)<len(B))
```

**(4 marks)**

**(c)** Let $S = \{1, 2, 4, 8, 16, 32\}$

**(i)** How many subsets are there of cardinality 4?

**(ii)** How many subsets of $S$ are there? That is, find $|\mathcal{P}(S)|$.

**(iii)** How many subsets of $S$ are there where the sum of the elements equals 13?

([$1+1+2$] **4 marks)**

**(Total 20 marks)**

## Question 2

**(a)** Which of the following well formed propositional formulas? Justify your answers.

**(i)** $\vee pq$ **(ii)** $p\neg\neg r$

**(iii)** $p\neg \rightarrow (q \wedge q)$ **(iv)** $(p \wedge \neg q) \vee (q\neg \rightarrow q)$

**(4 marks)**

**(b)** Consider the following code

```
message = 'Hi, I am 2.'
print('char', 'A', 'B', 'C', 'D', 'E', sep="\t")
for c in message:
    print(c, c.islower(), c.isupper(), c.isalpha(), c.isdigit(), sep="\t")
```

which produces the following table (only the first three rows are shown).

```
char    A       B       C       D
H       False   True    True    False
i       True    False   True    False
,       False   False   False   False

I

a
m

2
.
```

We have predicate

$$A(c) = \text{“character c is a lower case letter.”}$$

and similarly for $B(c)$, $C(c)$, and $D(c)$.
Complete the given table and determine the truth value of the following quantifiers. (Justify your answers)

**(i)** `all([c.isupper() for c in message])`

**(ii)** $\exists c\ \big[B(c) \wedge C(c)\big]$

**(iii)** $\forall c\ \big[C(c) \rightarrow (A(c) \vee B(c))\big]$

**(iv)** `any([c.isdigit() and c.isalpha() for c in message])`

$([4 + 4 \times 3]$ **16 marks)**

**(Total 20 marks)**

## Question 3

**(a)** Use a truth table to determine whether the proposition

$$((P \wedge Q) \vee (\neg P \wedge R)) \rightarrow (\neg Q \vee \neg R)$$

is a contradiction, is satisfiable or is a tautology. **(6 marks)**

**(b)** What is the output of the following code? (Justify your answer.)

```python
result = 0
for k in range(-5,5):
    term = k
    result += term
print(result)
```

**(4 marks)**

**(c)** Consider the sequence

$$7, 12, 17, 22, 27, \ldots$$

**(i)** Construct a recursive definition for the sequence.

**(ii)** Construct a closed formula for the $n$th term of the sequence.

**(iii)** Is 2023 a term in the sequence? Explain.

**(iv)** How many terms does the sequence $7, 12, 17, 22, 27, \ldots, 437$ have?

**(v)** Determine the sum: $7 + 12 + 17 + 22 + 27 + \cdots + 437$.

([5 × 2] **10 marks)**

**(Total 20 marks)**

## Question 4

**(a)** The function `is_what` has parameters $R$ (a relation), and $A$ and $B$ (both sets). What property does the function check for? (Justify your answer.)

```python
def is_what(R,A,B):
    for a,b in R:
        if a not in A:
            return False
        if b not in B:
            return False
    return {b for _,b in R}==B
```

**(5 marks)**

**(b)** Let $A = \{0, 2, 3\}$, $B = \{2, 3\}$, and $C = \{1, 5, 9\}$. Determine which of the following statements are true. (Justify your answers.)

**(i)** $5 \in C$ **(ii)** `{9} in C`

**(iii)** $\{3\} \subseteq A$ **(iv)** `A.intersection(B).issubset(B.union(C))`

**(v)** $B \subseteq A$ **(vi)** `B.union(C).intersection(A).isdisjoint(C)`

**(6 marks)**

**(c)** Evaluate each of the following.

**(i)** $\displaystyle\sum_{k=0}^{8} 2^k (k \bmod 2)$ **(ii)** $\displaystyle\sum_{k=1}^{5} (2k-1)$ **(iii)** $\displaystyle\prod_{k=0}^{8} (k+1)$

([3 × 3] **9 marks)**

**(Total 20 marks)**

## Question 5

**(a)** How many 6-bit strings (that is, bit strings of length 6) exist which satisfy each of the following criteria? (Justify your answers.)

**(i)** Start with the sub-string 01.

**(ii)** Have weight 3 (i.e., contain exactly three 1's) and start with the sub-string 01.

**(iii)** Either start with 01 or end with 11 (or both).

**(iv)** Have weight 3, starts with 01, and ends with 11.

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

**(b)** Let $R$ be a relation on the set $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ where $(a, b) \in R$ iff $a - b$ is a multiple of 3.

**(i)** Represent $R$ using a digraph.

**(ii)** Is $R$ reflexive? symmetric? transitive? (Justify your answer.)

**(iii)** Is $R$ an equivalence relation? and if yes, what are the resulting equivalence classes?

([3 + 3 + 2] **8 marks)**

**(c)** Consider the functions defined by the following Python code:

```python
def f(x):
    return 2*x + 3

def g(x):
    return x**3 - 1

def h(x):
    return 5 / x
```

Evaluate the following:

**(i)** $f(4)$ **(ii)** $g(2)$ **(iii)** $f(5) + g(2) - h(4)$

**(iv)** $f(g(1))$ **(v)** $g(h(2))$ **(vi)** $h(f(3))$

([6 × 1] **6 marks)**

**(Total 20 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

$$\begin{aligned}(p \leftrightarrow q) &\Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p) \\ &\Leftrightarrow (p \wedge q) \vee (\neg q \wedge \neg q)\end{aligned}$$

### Contrapositive

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

# Python Cheat Sheet

## Base Types

*integer, float, boolean, string, bytes*

| | | | | | |
|---|---|---|---|---|---|
| **int** | 163 | 0 | −192 | $\underbrace{\text{0b110}}_{\text{binary}}$ | $\underbrace{\text{0x3F}}_{\text{hex}}$ |
| **float** | 9.32 | 0.0 | $-1.7\underbrace{\text{E}-6}_{\times 10^{-6}}$ | | |
| **bool** | $\underbrace{\text{False}}_{0}$ | $\underbrace{\text{True}}_{1}$ | | | |
| **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,) | () |
| **str** | "153" | | | "" |

Immutable (non-modifiable values)

**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

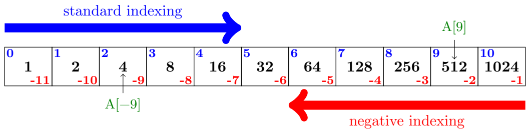
a.append(value)   a.extend(a2)
s.insert(idx,value)   a.pop()

## 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.