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

EXAMINATION:

DISCRETE MATHEMATICS (COMMON MODULE)

SEMESTER 1 - YEAR 1 - REPEAT

AUGUST 2024 DURATION: 2 HOURS

INTERNAL EXAMINERS: DR KIERAN MURPHY
DR DENIS FLYNN

DATE: TUE, 20 AUGUST 2024
TIME: 11:45 AM
VENUE: MAIN HALL

EXTERNAL EXAMINER: MS MARGARET FINNEGAN

INSTRUCTIONS TO CANDIDATES

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

MATERIALS REQUIRED

1. NEW MATHEMATICS TABLES.
2. GRAPH PAPER

SOUTH EAST TECHNOLOGICAL UNIVERSITY

Question 1

(a) Consider the sequence 5, 8, 11, 14, 17, 20, ...

(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)

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

```
def isWhat(n):
  if (n > 2):
    for i in range(2, n//2+1):
      if (n % i) == 0:
        return False
      else:
        continue
    return True
  elif (n==2):
    return True
  else:
    return False
```

(4 marks)

(c) How many shortest lattice paths start at $(2, 3)$ and

(i) end at $(7, 12)$?

(ii) end at $(7, 12)$ and pass through $(4, 8)$?

(iii) end at $(7, 12)$ and avoid $(4, 8)$?

([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 - 4*x - 5
def g(x):
        return x + 5
def h(x):
        return x - 5
def j(x):
    return x//2
def k(x):
    return 2*x
```

Evaluate the following (note, show all work):

(i) $g(h(4))$ **(ii)** $h(g(5))$ **(iii)** $f(g(h(3)))$ **(iv)** $k(j(7))$ **(v)** $j(j(g(2)))$

(5 marks)

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

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

is satisfiable.

(5 marks)

(c) Evaluate each of the following

(i) $\sum_{k=1}^{5} k + 2$ **(ii)** $\sum_{k=0}^{4} 2^k$ **(iii)** $\prod_{k=2}^{5} (k-2)$

(6 marks)

(d) Which of the following are well formed propositional formulas?

(i) $p\neg q \wedge r$ **(ii)** $\neg(q \wedge r) \to s$ **(iii)** $\to (p \to \neg s)$ **(iv)** $p \leftrightarrow (p \to s\neg)$

Justify your answers.

(4 marks)

Question 3

(a) Consider the sets defined by the Python code

```
U = set(range(1, 10))
A = set(range(1, 8, 2))
B = set(range(2, 9, 2))
C = set(range(3, 6))
                                                    ;
D = A.intersection(C).union(U.difference(B))
E = U.difference(A.union(B).intersection(C))
F = A.union(U.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 $S = \{1, 2, 3, 4, 5, 6\}$

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

(ii) How many subsets of cardinality 4 have $\{2, 3, 5\}$ as a subset?

(iii) How many subsets of cardinality 4 contain at least one odd number?

(iv) How many subsets of cardinality 4 contain exactly one even number?

($[4 \times 2]$ 8 marks)

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

(6 marks)

Question 4

(a)

```
A = set(range(1,5))
R = {(a, b) for a in A for b in A if abs(a - b) <= 2}
```

(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).

($[3+3+3+2+3]$ 14 marks)

(b) How many 14-bit strings (that is, bit strings of length 14) are there which satisfy each of the following criteria? Explain your answer.

(i) End with the sub-string 0011.

(ii) Have weight 7 (i.e. contain exactly seven 1's in total) and end with the sub-string 0011.

(iii) Have weight of 7 and are divisible by 16.

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

Question 5

(a) In a Python program, the variable x stores an unknown value.
Running the following Python code (where % denotes mod and // denotes integer division)

```
print (x%9, x//8)
```

produces the output:

```
2,4
```

Determine a value for x that produced the above output.

(4 marks)

(b) Consider the following code:

```
password = 'AaBb12#$'
print('char', 'A', 'B', 'C', 'D', sep="\t")
for c in password :
    print(c, c.isdigit(), c.isupper(),c.islower(), c.isalpha(), sep="\t")
```

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

char	D	U	L	A
A	False	True	False	True
a	False	False	True	True
B	False	True	False	True
b				
1				
2				
#				
$				

We have predicate

$$D(c) = \text{“Character c is a digit.”}$$

and similarly we have predicates for $U(c), L(c)$, and $A(c)$.

Complete the given table and determine the truth value of the following quantifiers.

(i) **any**([c. isdigit () **for** c **in** password])

(ii) **all** ([c. isalpha **or** c. isdigit () **or** c.isupper() **for** c **in** password])

(iii) $\nexists c \left[U(c) \wedge L(c)\right]$

(iv) $\exists c \left[\neg U(c) \wedge \neg L(c) \wedge \neg D(c) \wedge \neg A(c)\right]$

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

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

Indirect Reasoning (Modus Tollens)

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

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

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

Resolution

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

Conditional Equivalence

$$p \to q \Leftrightarrow \neg p \vee q$$

Biconditional Equivalences

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

Contrapositive

$$p \to q \Leftrightarrow \neg q \to \neg p$$

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

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()

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'

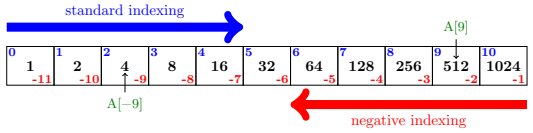
(Convert each element in a collection)
[**int**(x) **for** x **in** ['1','29','−3']] → [1, 29, −3]

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.