PAPER CODE NO.	EXAMINER:	Dr. Will Rosenbaum	Tel. No. XXX-XXXX
COMP526	DEPARTMENT:	Computer Science	

UNIVERSITY OF LIVERPOOL

FIRST SEMESTER EXAMINATIONS 2023/24

Efficient Algorithms

– with model solutions –

TIME ALLOWED : TWO AND A HALF Hours

INSTRUCTIONS TO CANDIDATES

The exam mark is the sum of the **best 4 out of 5** questions; four questions are required.

If you attempt to answer more questions than the required number of questions, the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).

Each task states the **expected parts of your answer**; this list is comprehensive, i.e., only the stated parts are required for full marks.

The only allowed material is a single A4 sheet of paper with your own handwriting notes on both sides.

Calculators are NOT permitted.

Question 1

1.A What is the order of growth of the **worst-case** running time of **Mergesort**? **(5 marks)**

(a) $\Theta(\log n)$	**(c)** $\Theta(n)$	**(e)** $\Theta(n \log^2 n)$	**(g)** $\Theta(n^2)$
(b) $\Theta(\sqrt{n})$	**(d)** $\Theta(n \log n)$	**(f)** $\Theta(n \cdot \sqrt{n})$	**(h)** $\Theta(n^3)$

Solution for 1.A **[Reproduction/Bookwork (maybe with simple application)]**

$\Theta(n \log n)$.

1.B **Decode** the **Run-Length Encoded** bitstring **(10 marks)**

```
10001101011001000100000010011
```

Make sure you clearly indicate your **final answer**!
While it is sufficient to give the final answer, in case your answer is not correct, part marks can be given for well-documented *intermediate steps* showing how you arrived at the answer.

Solution for 1.B **[Application of known method on new examples]**

We first find the first bit 1. Then, we decode the *Elias Gamma coded* run lengths; this yields

$$13, 3, 4, 2, 19.$$

We now generate runs of these lengths, alternating between 0 and 1, starting with 1. So the answer is

```
11111111111110001111001111111111111111111
```

1.C Let $O = (0, 2, 4, 0, 3, 4, 2, 5, 4, 2, 0, 4, 4)$ be the output sequence of indexes of the **move-to-front (MTF) transform** on text T with initial queue content (alphabet) $Q = [\mathtt{B}, \mathtt{E}, \mathtt{I}, \mathtt{R}, \mathtt{T}, \mathtt{W}]$. Decode the text T. **(10 marks)**

Expected parts of your answer:

(i) *state of Q after each step*
(ii) *original text T*

total: 10 marks

Solution for 1.C **[seen with different values]**

(i)

O		*Q* (new)		*T*[*i*]
(**0**,2,4,0,3,4,2,5,4,2,0,4,4)	–	[B,E,I,R,T,W]	–	B
(0,**2**,4,0,3,4,2,5,4,2,0,4,4)	–	[I,B,E,R,T,W]	–	I
(0,2,**4**,0,3,4,2,5,4,2,0,4,4)	–	[T,I,B,E,R,W]	–	T
(0,2,4,**0**,3,4,2,5,4,2,0,4,4)	–	[E,T,I,B,R,W]	–	T
(0,2,4,0,**3**,4,2,5,4,2,0,4,4)	–	[E,T,I,B,R,W]	–	E
(0,2,4,0,3,**4**,2,5,4,2,0,4,4)	–	[R,E,T,I,B,W]	–	R
(0,2,4,0,3,4,**2**,5,4,2,0,4,4)	–	[T,R,E,I,B,W]	–	T
(0,2,4,0,3,4,2,**5**,4,2,0,4,4)	–	[W,T,R,E,I,B]	–	W
(0,2,4,0,3,4,2,5,**4**,2,0,4,4)	–	[I,W,T,R,E,B]	–	I
(0,2,4,0,3,4,2,5,4,**2**,0,4,4)	–	[T,I,W,R,E,B]	–	T
(0,2,4,0,3,4,2,5,4,2,**0**,4,4)	–	[T,I,W,R,E,B]	–	T
(0,2,4,0,3,4,2,5,4,2,0,**4**,4)	–	[E,T,I,W,R,B]	–	E
(0,2,4,0,3,4,2,5,4,2,0,4,**4**)	–	[R,E,T,I,W,B]	–	R

(ii) *T* = BITTERTWITTER

UNIVERSITY OF LIVERPOOL

Question 2

2.A Consider the two functions **(5 marks)**

$$f(n) = 84\sqrt{n} + 31n\log(\log(n)) + \frac{4n}{\log(n)} \tag{1}$$
$$g(n) = 24n^{3/2} + 66n + 43 \cdot 2^{-n} \tag{2}$$
$$\tag{3}$$

How do $f(n)$ and $g(n)$ compare as $n \to \infty$?

(a) $f = o(g)$

(b) $f = \Theta(g)$

(c) $f = \omega(g)$

Solution for 2.A **[Reproduction/Bookwork (maybe with simple application)]**

The functions are sums of simpler functions; using the known asymptotic relations between those, we easily find

$$f(n) \sim 31n\log(\log(n)) \tag{4}$$
$$g(n) \sim 43 \cdot 2^{n} \tag{5}$$
$$\tag{6}$$

and thus

$$f(n) = o(g(n)) \qquad (n \to \infty).$$

The correct answer is

1

2.B Given the 7-bit block $B = B_7 \dots B_1 = 1111001$ received as the block of a **(7,4) Hamming code**. **(10 marks)**

Decode the 4-bit message $D = D_3D_2D_1D_0$ from B and whether a corrected 1-bit error has occurred.

Your answer should give the four bits $D_3D_2D_1D_0$ as well as “error occurred” resp. “no error occurred”.

Make sure you clearly indicate your **final answer**!

While it is sufficient to give the final answer, in case your answer is not correct, part marks can be given for well-documented *intermediate steps* showing how you arrived at the answer.

Solution for 2.B [Application of known method on new examples]

We first compute the checking bits: $C = C_2C_1C_0 = 001$; as a binary number this is $c = 1$, which indicates the position of a error (0 if none).

If $c \neq 0$, we flip bit B_c and then extract the message from it; in our case $D_3D_2D_1D_0 = 1110$.

So, the correct answer is

```
1110Y
```

2.C Suppose the following keys are inserted in the given order into an (initially empty, unbalanced) **binary search tree**: **(10 marks)**

$$55, 53, 76, 26, 14, 43, 51, 41, 50, 10, 23, 20, 83, 29, 1$$

In the resulting binary search tree, we now search for the key 83.

What keys is the sought key compared to? Enter those keys in the order they are compared to 83, separated by commas.

Make sure you clearly indicate your **final answer**!

While it is sufficient to give the final answer, in case your answer is not correct, part marks can be given for well-documented *intermediate steps* showing how you arrived at the answer.

Solution for 2.C [Application of known method on new examples]

The search starts at the root and then proceeds down the tree: left if the sought key is strictly smaller than the node's key, right if it is strictly larger, and terminates successfully if they node's key is equal to the sought key. If we reach null, the search terminates unsuccessfully (the sought key is not stored in the tree).

Applying this procedure here yields

$$55, 76, 83$$

Question 3

3.A Briefly explain the high-level idea how to find all k **occurrences** of a **pattern** $P[0..m-1]$ in a text $T[0..n-1]$ using the *suffix array* R of the text. **(5 marks)**

State the time complexity of this procedure.

Expected parts of your answer:

(i) *description of algorithm (1-3 sentences)*
(ii) *Θ-class of running time complexity*

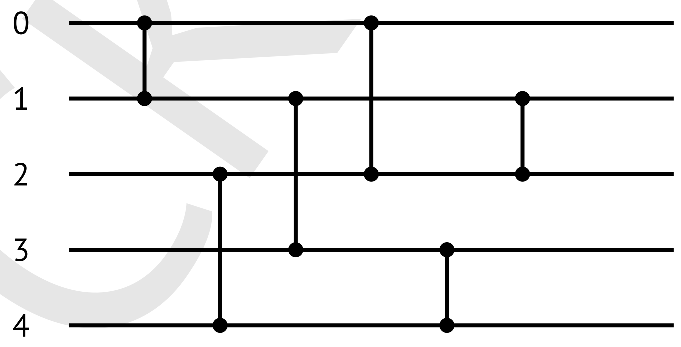
total: 5 marks

Solution for 3.A **[bookwork]**

(i) Use 2 binary searches on R where we compare a position i in the text with the pattern P by lexicographically comparing $T[i..i+m]$ with $P \cdot \#$, where # is once treated smaller than all characters in the alphabet and once treated larger. That gives us the interval in R containing the starting positions of all occurrences of P in T.

(ii) $\Theta(m \log n + k)$

3.B Consider the following comparator network on 5 wires: **(10 marks)**



(a) How many comparators are in the network?
(b) What is the depth of the network?
(c) What is the output of the network on input $[4, 3, 1, 2, 5]$?
(d) Is the network a sorting network? Explain why or why not.

Solution for 3.B **[Application of known method on new examples]**

(a) There are 6 comparators.
(b) The depth of the network is 3.
(c) The output is $[1, 2, 3, 4, 5]$.

(d) The network is not a sorting network. For example, there is no directed path from wire 3 to wire 0, so if the smallest value is initially at index 3, after applying the comparator network, the smallest value cannot be on wire (index) 0.

3.C You are tasked with implementing an **anagram finder**. Specifically, you are given a list of words $W_1, \dots, W_n$, each m characters long. Your task is to output a **pair of words**, W_i and W_j, so that W_j is an **anagram** of W_i, i.e., W_j can be formed from W_i by permuting its letters – or "no anagrams" in case no such pair exists. (No letters are added or removed, only rearrangement is allowed in anagrams; for example `santa` is an anagram of `satan`, and `supersonic` is an anagram of `percussion`). **(10 marks)**

We assume that the W_i are English words over the usual alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \dots, \texttt{z}\}$ and that their length m is moderate (maybe between 10 and 20 characters). On the other hand, the number of words n could be huge.

Describe an efficient algorithm for this problem.

You should (i) describe the main idea in one sentence, (ii) give the algorithm in pseudocode (where you can use algorithms from the lecture) and (iii) state the required running time of your algorithm.

Solution for 3.C **[Transfer task, requires some insight]**

(i) we sort the words to identify anagrams, and sort the list of sorted words to find duplicates

(ii)

```
(1) For each $W_i$, create an object $X[i] = (W_i, W_i)$
(2) For each $i$, sort the letters in $X[i][0]$, the first entry of the object.
(3) Sort $X$ by the first entry of the pair.
(4) For $i = 0, \dots, n-2$:
        i. if $X[i][0] = X[i+1][0]$, return $(X[i][1], X[i+1][1])$
```

(iii) Step (1) takes $\Theta(n \cdot m)$ time (if we assume that we create copies of the words).

Step (2) takes time $\Theta(n \cdot m \log m)$ time if implemented using a general purpose sorting method, but it can be implemented in $\Theta(nm)$ time using counting sort.

Step (3) uses $\Theta(n \log n \cdot m)$ time with a general purpose sorting algorithm since each comparison involves comparing two strings of length m. However, we can use fat-pivot radix quicksort instead (or another radix sort) to sort in $\Theta(nm)$ total time.

Step (4) takes $O(nm)$ time for the character comparisons.

Overall, the algorithm can be implemented in time linear in the total number of characters.

Question 4

4.A Suppose P and Q are logical predicates. Compute the logical negation of the expression $(\exists x)(\forall y)[P(x) \implies Q(y)]$. **(5 marks)**

Solution for 4.A **[seen with different values]**

$(\forall x)(\exists y)[P(x) \wedge \neg Q(y)]$

4.B Assume you have 5 symbols A, B, C, D and E forming an alphabet Σ, i.e., $\Sigma = \{A, B, C, D, E\}$. The distribution of probabilities (i.e. weights) on symbols in the alphabet Σ is as follows: $P(\mathrm{A}) = 0.10$, $P(\mathrm{B}) = 0.20$, $P(\mathrm{C}) = 0.40$, $P(\mathrm{D}) = 0.25$, and $P(\mathrm{E}) = 0.05$. Construct, step by step, the **Huffman tree** for symbols in Σ and list the codes for symbols A, B, C, D and E. **(10 marks)**

Strictly follow these conventions for constructing the code: To break ties, first use the smallest letter (according to the alphabetical order), or the tree containing the smallest alphabetical letter. When combining two trees of different values, place the lower-valued tree on the left (corresponding to a 0-bit). When combining trees of equal value, place the one containing the smallest letter to the left.

Further, state the time complexity of the construction and compute the bit rate of the obtained code.

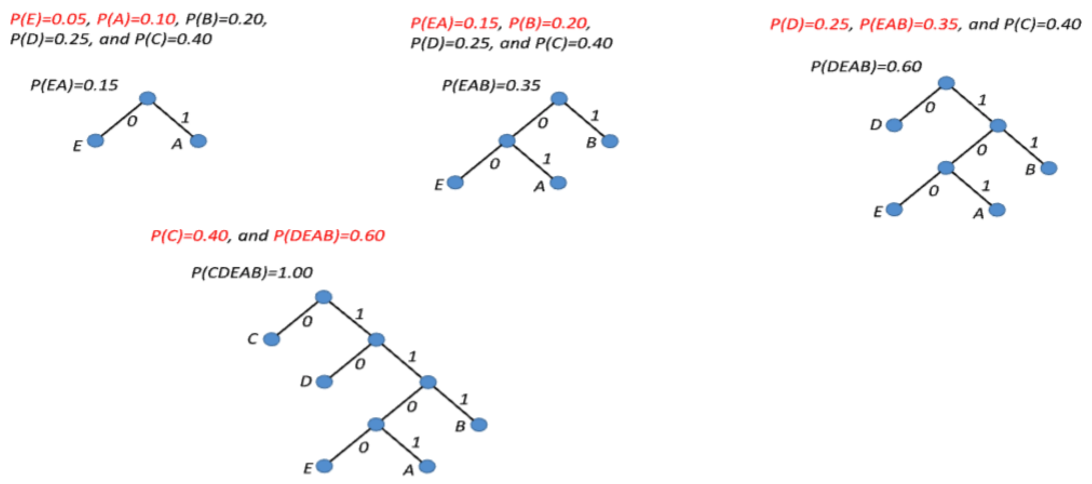
Expected parts of your answer:

(i) *(intermediate) trees after each step*
(ii) *codewords for all symbols*
(iii) *bit rate of the code*
(iv) *Θ-class of running time*

total: 10 marks

Solution for 4.B **[seen with different values]**

(i)


P(E)=0.05, P(A)=0.10, P(B)=0.20, P(D)=0.25, and P(C)=0.40
P(EA)=0.15
P(EA)=0.15, P(B)=0.20, P(D)=0.25, and P(C)=0.40
P(EAB)=0.35
P(D)=0.25, P(EAB)=0.35, and P(C)=0.40
P(DEAB)=0.60
P(C)=0.40, and P(DEAB)=0.60
P(CDEAB)=1.00

(ii) We obtain the codes: A = `1101`, B = `111`, C = `0`, D = `10`, and E = `1100`

(iii) 2.1 bit/symbol

(We compute $4 \cdot 0.10 + 3 \cdot 0.20 + 1 \cdot 0.40 + 2 \cdot 0.25 + 4 \cdot 0.05 = 2.1$.)

(iv) $\Theta(n \log n)$

During each step of the algorithm we have to maintain access to two groups with the smallest weight. This operation is implementable in logarithmic time with the help of a priority queue. Thus if the size of the alphabet is n, the total cost of Huffman tree construction is $O(n \log n)$.)

4.C Given the pattern $P = P[0..m]$ = `dbbabdbbabbc` with $m = 12$ over alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$. Compute the **string-matching DFA** for P. **(10 marks)**

List the automaton by giving the transition function $\delta(q, c)$ for all states q and characters c. More specifically, your answer should give a **table** with one row for each symbol $c \in \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$ and one column for each state $q \in [0..12)$.

Make sure you clearly indicate your **final answer**!

While it is sufficient to give the final answer, in case your answer is not correct, part marks can be given for well-documented *intermediate steps* showing how you arrived at the answer.

Solution for 4.C **[Application of known method on new examples]**

d(c,q)	0	1	2	3	4	5	6	7	8	9	10	11
a	0	0	0	4	0	0	0	0	9	0	0	0
b	0	2	3	0	5	0	7	8	0	10	11	0
c	0	0	0	0	0	0	0	0	0	0	0	12
d	1	1	1	1	1	6	1	1	1	1	6	1

Question 5

5.A Explain the basic operations defined for the **stack** ADT and state their time complexities when implemented using a linked list. **(5 marks)**

Expected parts of your answer:

- ***(i)*** *3 operations (1 sentence per operation)*
- ***(ii)*** *Θ-class for each operation*

total: 5 marks

Solution for 5.A **[bookwork]**

(i)
push(x): add x on top of the stack
pop(): remove and return the topmost element on the stack
isEmpty(): return whether the stack is empty

(ii) all operations require Θ(1) time.

5.B Given text T = `abaabaaab$`, draw both both the standard **trie** containing all suffixes of T as well as the **suffix tree** of T. Finally, compute the **suffix array** L for T. **(10 marks)**

Comment briefly on the space usage of suffix trees and suffix arrays (for strings built over large, but constant-size alphabets).

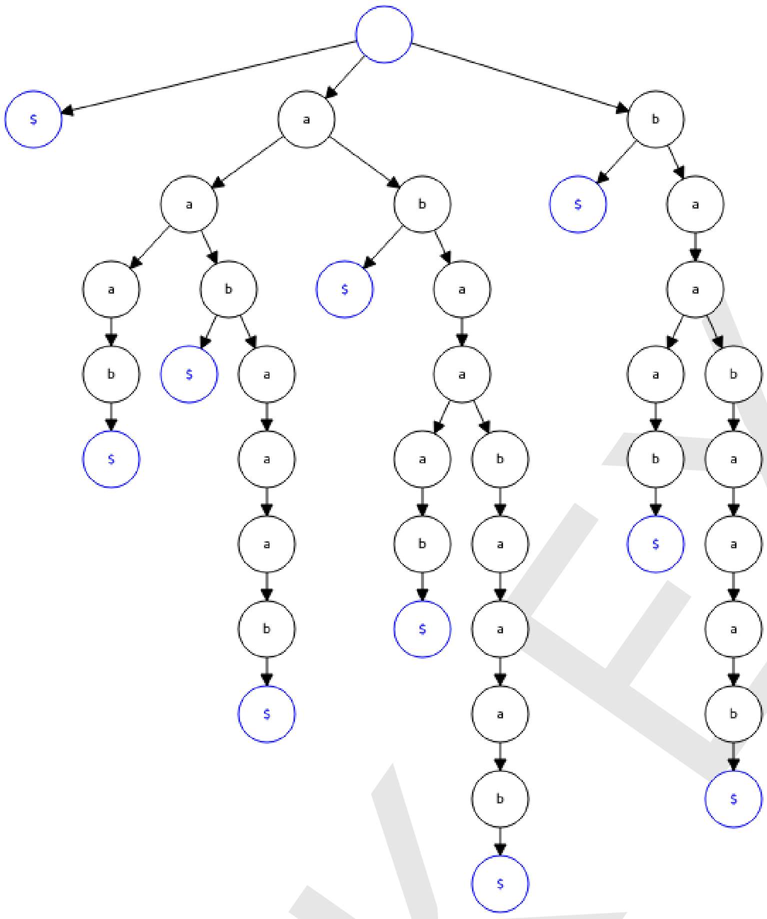
Expected parts of your answer:

- ***(i)*** *trie of suffixes*
- ***(ii)*** *suffix tree (conceptual version with strings on edges)*
- ***(iii)*** *suffix array*
- ***(iv)*** *state Θ-class of worst-case size*
- ***(v)*** *comment on constants in size*

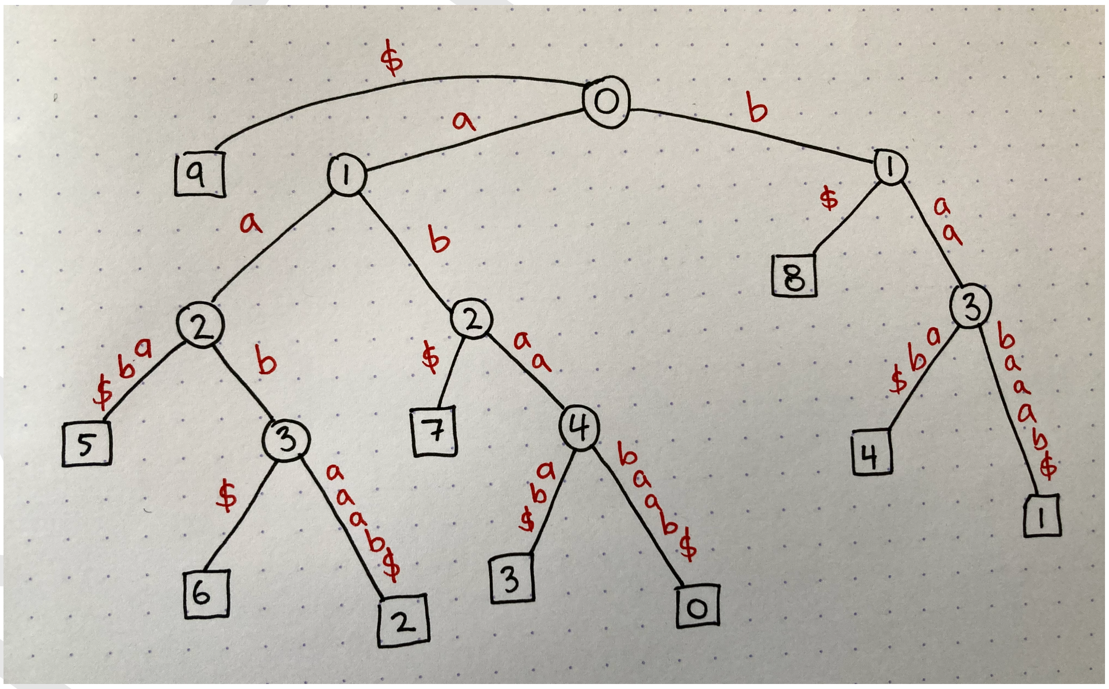
total: 14 marks

Solution for 5.B **[seen with different values,bookwork]**

(i)



(ii)



(iii) $L[0..9] = [9, 5, 6, 2, 7, 3, 0, 8, 4, 1]$.

(iv) Both suffix array and suffix tree use $\Theta(n)$ words (characters) of space.

(v) The suffix array (without additional data structures) requires exactly n words, whereas the suffix tree has to store σ pointers for each of the n nodes, plus the labels of edges, so its space usage has a much larger constant factor.

5.C Design an efficient algorithm for the **closest numbers** problem: **(10 marks)**

Given: An array of floating-point numbers $A[0..n)$.

Goal: Return indices i, j with the minimal distance $|A[i] - A[j]|$ between numbers $A[i]$ and $A[j]$.

You should (i) describe the main idea in one sentence, (ii) give the algorithm in pseudocode (where you can use algorithms from the lecture) and (iii) state the required running time of your algorithm.

Solution for 5.C **[Transfer task, requires some insight]**

(i) We sort A; then the closest pair must be adjacent.

(ii)

```
B[0..n) = new array
for i = 0, ..., n − 1
    B[i] := (A[i], i)
sort(B[0..n))
minDiff = +∞
argMin = −1
for i = 0, ..., n − 2
    if B[i + 1][0] − B[i][0] < minDiff
        minDiff = B[i + 1][0] − B[i][0]
        argMin = i
return (B[i][1], B[i + 1][1])
```

(iii) Time is dominated by sorting $\Theta(n \log n)$.

UNIVERSITY OF LIVERPOOL

Marks for Incomplete and Incorrect Solutions

Marks for incomplete and/or incorrect solutions of sub-tasks are based on the marking descriptor of our MSc programmes. The 'percent' values are to be read as the respective fraction of the marks, the marks assigned will be rounded. For example, 75% out of 6 marks give a raw mark of 4.5, which is rounded to 5, while 70% out of 6 marks gives a raw mark of 4.2, which is rounded to 4.

The marking descriptor used for determining raw marks, which follows the description provided in the MSc programme specifications, is:

Distinction grades:

$\geq$ **80%:** Factually almost faultless; clearly directed; logical; comprehensive coverage of topic; substantial elements of originality and independent thought; very well written.

70–80%: Logical; enlightening; originality of thought or approach; good coverage of topic; clear, in-depth understanding of material; very well written and directed.

Pass grades:

60–70%: Logical; thorough; factually sound (no serious errors); good understanding of material; exercise of critical judgement; some originality of thought or approach; well written and directed.

50–60%: Worthy effort, but undistinguished outcome. Essentially correct, but possibly missing important points. Some evidence of critical judgement; some weaknesses in expression or presentation.

Fail grades:

40–50%: Incomplete coverage of topic; evidence of poor understanding of material; Poor presentation; lack of coherent argument.

$\leq$ **40%:** Serious omissions; significant errors/ misconceptions; poorly directed at targets; evidence of inadequate effort.