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

UNIVERSITY OF LIVERPOOL

FIRST SEMESTER EXAMINATIONS 2023/24

Efficient Algorithms

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)** $\Theta(n^3)$

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.

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) $Q = [\texttt{B}, \texttt{E}, \texttt{I}, \texttt{R}, \texttt{T}, \texttt{W}]$. Decode the text T. **(10 marks)**

Expected parts of your answer:

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

Question 2

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

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

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

(a) $f = o(g)$

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

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

2.B Given the 7-bit block $B = B_7 \ldots B_1 = 1111001$ 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.

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.

Question 3

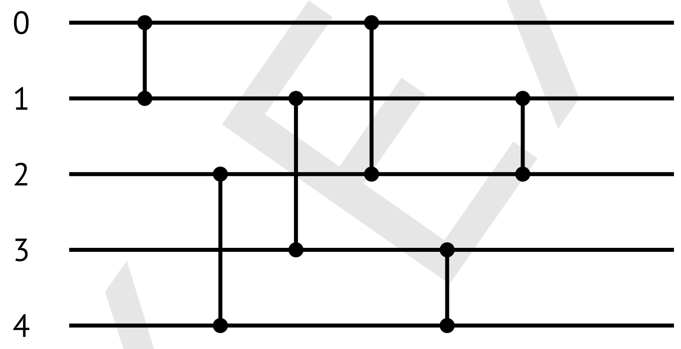
3.A Briefly explain the high-level idea how to find all k **occurrences** of a **pattern** $P[0..m-1]$ in a text $T[0..n-1]$ using the *suffix array* R of the text. **(5 marks)**

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*

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.

3.C You are tasked with implementing an **anagram finder**. Specifically, you are given a list of words $W_1, \ldots, W_n$, 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 $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots, \mathtt{z}\}$ 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.

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)**

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: $P(A) = 0.10$, $P(B) = 0.20$, $P(C) = 0.40$, $P(D) = 0.25$, and $P(E) = 0.05$. 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*

4.C Given the pattern $P = P[0..m) = \texttt{dbbabdbbabbc}$ 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.

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*

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*

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.