Cambridge AS & A Level · Computer Science 9618

Computer Science 9618 formulas and definitions: A2

All 84 formulas, facts and definitions to learn, by syllabus topic. Formulas the exam gives you are left out.

Read online at **pastopic.com/cambridge/computer-science-9618/formulas**; each topic heading links to its full note.

© 2026 Pastopic. Free for personal study. Please share the link, not the file.

A Level

Data representation

Enumerated type: `TYPE <name> = (<value1>, <value2>, …)`

Pointer type: `TYPE <name> = ^<data type>`; address of `X`: `^X`; value pointed to: `P^`

Record type: `TYPE <name>` … `DECLARE <field> : <type>` … `ENDTYPE`

Record variable: `DECLARE <var> : <record type>`; field: `<var>.<field> ← <value>`

Set: `TYPE <name> = SET OF <type>` then `DEFINE <set> (<values>) : <name>`

Floating-point value $= M \times 2^E$ (mantissa a two's complement fraction, point after the first bit; exponent a two's complement integer)

Mantissa place values: $-1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$; exponent with e bits: $-2^{e-1}, \ldots, 2, 1$

Normalised mantissa: positive starts 01, negative starts 10

Moving the mantissa one place left: exponent -1

m-bit mantissa, e-bit exponent: largest $= \left(1 - 2^{-(m-1)}\right) \times 2^{2^{e-1}-1}$

Smallest positive normalised $= \frac{1}{2} \times 2^{-2^{e-1}}$

More mantissa bits → more precision; more exponent bits → more range

Hashing: address = key `MOD` n (for n addresses 0 to $n - 1$)

Communication and internet technologies

Protocol = an agreed set of rules for communication, so different hardware and software can exchange data

TCP/IP layers, top to bottom: Application, Transport, Internet, Link

Each layer only passes data to the layer directly above or below it; sending goes down the stack, receiving goes up

Transport: segments, sequence numbers, ports, end-to-end delivery, flow control, resending

Internet: IP addresses in the header (datagrams), routing

Link: frames, MAC addresses, the physical link to the next device

HTTP(S): web pages · FTP: files · SMTP: send email (push) · POP3 and IMAP: receive email (pull) · BitTorrent: peer-to-peer file sharing

POP3 downloads to one device; IMAP keeps mail on the server and synchronises every device

BitTorrent: torrent file (metadata) → tracker → swarm of peers → pieces → seeds

Circuit switching: dedicated path set up first, whole bandwidth, same route, in order

Packet switching: packets with headers, independent routes, reordered by sequence number, lost packets resent

Hop count: reduced by 1 at each router; the packet is discarded at 0

Hardware and virtual machines

RISC: few, simple, fixed-length instructions; one clock cycle each; few addressing modes; many general-purpose registers; hard-wired control unit; easy to pipeline; emphasis on software

CISC: many complex, variable-length instructions; several clock cycles each; many addressing modes and formats; microprogrammed control unit; uses cache; emphasis on hardware

Pipeline stages: IF, ID, OF, IE, WB; n instructions in a 5-stage pipeline take $n + 4$ cycles (not $5n$)

SISD, SIMD, MISD, MIMD: single or multiple **instruction** streams, then single or multiple **data** streams

Half adder: $S = A \oplus B, C = A.B$

Full adder: $S = A \oplus B \oplus C_{in}$ (1 for an odd number of 1s); $C_{out} = 1$ when two or more inputs are 1

SR: 10 sets, 01 resets, 00 holds, 11 invalid. JK: the same, but 11 toggles, and it is clocked

De Morgan: $\overline{A.B} = \overline{A} + \overline{B}$ and $\overline{A + B} = \overline{A}.\overline{B}$

$A + \overline{A} = 1, A.\overline{A} = 0, A + A.B = A, A + \overline{A}.B = A + B$

K-map order $00, 01, 11, 10$; loops of $1, 2, 4, 8$; wrap around the edges

System software

Process states: running, ready, blocked. Ready → running (given the processor); running → ready (time slice ends or pre-empted); running → blocked (waits for I/O or event); blocked → ready (event complete)

FCFS: arrival order, to the end. SJF: shortest burst first, to the end (non-pre-emptive). SRT: shortest time left, pre-emptive. Round robin: equal time slices in turn, pre-emptive

Kernel as interrupt handler: receive, check priority, save state, find ISR in the interrupt dispatch table, run it, restore state

Paging: fixed-size pages and frames, page table, page number + offset. Segmentation: variable-size logical segments, segment map table, segment number + offset

Page replacement: FIFO (oldest page) or LRU (least recently used page)

Disk thrashing: pages swapped in and out so often that more time is spent swapping than processing

Compilation: lexical analysis → syntax analysis → code generation → optimisation

BNF: `::=` is defined as, | or, `<name>` non-terminal; repetition by recursion, such as `<digits> ::= <digit> | <digit><digits>`

RPN with a stack: push operands; for an operator pop two, apply (second popped on the left), push the result

Security

Plaintext → (encrypt with a key) → ciphertext → (decrypt with a key) → plaintext

Symmetric: one shared secret key encrypts and decrypts; fast, shorter keys (128/256 bits), but the key must be shared

Asymmetric: a matching public/private key pair; one encrypts, only the other decrypts; slower, longer keys (2048 bits), only the public key is shared

Private message: encrypt with the **receiver's public key**, decrypt with the **receiver's private key**

Verified message: encrypt with the **sender's private key**, decrypt with the **sender's public key**

Digital signature = message digest (hash of the message) encrypted with the sender's private key

Check: decrypt signature with sender's public key → digest 1; hash received message → digest 2; equal means unaltered

Digital certificate: issued by a CA; holder's name and public key, serial and version numbers, validity dates, issuer, CA's signature

Quantum cryptography: photons over fibre; eavesdropping detected because measuring changes the state

SSL/TLS: encryption, authentication, integrity; handshake protocol + record protocol

TLS handshake: certificate sent → verified → server's public key → algorithms agreed → symmetric session keys → secure session

Artificial intelligence (AI)

Graph: **nodes** (vertices) joined by **edges**; edges can be **weighted** (distance, time, cost)

Dijkstra's: shortest distance from the start to **every** node; set start $= 0$, others $= \infty$; visit the smallest unvisited; update a neighbour only if the new total is smaller

A*: $f = g + h$, where g = actual cost from the start, h = heuristic (estimated cost to the goal); expand the lowest f

A* uses a heuristic to find the shortest path between **two** nodes, exploring fewer nodes; Dijkstra's explores all routes

ANN: input layer → hidden layers → output layer; weighted connections; weights adjusted by training

Back propagation: compare actual with expected output → error gradient (calculus) → feed error back → adjust weights → repeat

Supervised = **labelled** data, predicts outputs; unsupervised = **unlabelled** data, finds hidden patterns/clusters; reinforcement = **rewards and penalties**, trial and error

Deep learning = machine learning with ANNs that have **many hidden layers**; best with large data sets

Regression: predicts a **numerical value** by fitting a line or curve that minimises the errors

Computational thinking and problem-solving

Linear search: worst case n comparisons, $O(n)$

Binary search: worst case $\lfloor \log_2 n \rfloor + 1$ comparisons, $O(\log_2 n)$; data must be sorted

Bubble sort and insertion sort: $O(n^2)$ worst case, $O(n)$ if already sorted, space $O(1)$

Stack push and pop, queue enqueue and dequeue: $O(1)$

Middle index: `Mid ← (Lower + Upper) DIV 2`

Circular wrap: `IF Pointer > MaxSize THEN Pointer ← 1` (or `0` for a 0-based array)

Stack: LIFO, one end, one moving pointer (Top)

Queue: FIFO, two ends, two moving pointers (Front, Rear)

Recursion: general case, base case, moves towards the base case

Further programming

Immediate: operand is the value. Direct: operand is the address. Indirect: operand is the address of the address. Indexed: operand + IX. Relative: offset from a base address

Class: template of attributes and methods. Object: an instance of a class

Encapsulation: data and methods in one class, attributes `PRIVATE`, reached through methods

Getter returns an attribute's value; setter changes it

Inheritance: subclass takes the superclass's attributes and methods and can add or override them

Polymorphism: same method name, redefined to behave differently in subclasses

Containment (aggregation): a class holds objects of another class ("has a")

Fact `f(a, b).`, rule `head(X) IF p(X) AND NOT q(X).`, goal `f(X, b)` returns `X = …`

`OPENFILE … FOR READ / WRITE / APPEND / RANDOM`, `READFILE`, `WRITEFILE`, `EOF()`, `CLOSEFILE`

`SEEK file, position`, `GETRECORD file, variable`, `PUTRECORD file, variable`

Exception: unexpected run-time event; causes: programming, user, hardware errors