

Computer Science 0478 formulas and definitions

All 268 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-0478/formulas; each topic heading links to its full note.

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

1.1 Number systems

Denary base 10 (digits 0–9); binary base 2 (0 and 1); hexadecimal base 16 (0–9 and A–F)

8-bit place values: 128, 64, 32, 16, 8, 4, 2, 1 (keep doubling for 12 or 16 bits)

Hex place values: 256, 16, 1; A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

1 hex digit = 4 bits = 1 nibble; 8 bits = 1 byte = 2 hex digits

Largest value in n bits = $2^n - 1$ (8 bits: 255 = FF; 12 bits: 4095 = FFF; 16 bits: 65 535 = FFFF); smallest = 0

Binary addition: $0 + 0 = 0$, $0 + 1 = 1$, $1 + 1 = 10$, $1 + 1 + 1 = 11$

Overflow: the result is greater than the largest value the register can hold (above 255 in 8 bits)

Logical left shift of n places = $\times 2^n$; logical right shift of n places = $\div 2^n$; lost bits are gone and 0s fill the gap

Two's complement 8-bit: MSB worth -128 ; range -128 to $+127$

Negative denary $\rightarrow$ two's complement: positive in binary, flip the bits, add 1

1.2 Text, sound and images

Character set: every character has a unique binary code; text is stored as the codes of its characters, in order

ASCII: 7 bits per character, 128 characters (extended ASCII: 8 bits, 256); A = 65, a = 97, codes in alphabetical order

Unicode: more bits per character (up to 32), far more characters (other languages, symbols, emojis), so more storage

Sample rate = number of samples per second (Hz); sample resolution = number of bits per sample

Resolution = number of pixels (width $\times$ height); colour depth = number of bits per colour

n bits give 2^n different values (colours, or amplitude levels)

Higher sample rate / sample resolution / resolution / colour depth $\rightarrow$ more accurate or better quality, and a larger file

Image file size (bits) = width $\times$ height $\times$ colour depth

Sound file size (bits) = sample rate $\times$ sample resolution $\times$ seconds

1 nibble = 4 bits; 1 byte = 8 bits; 1 KiB = 1024 bytes; 1 MiB = 1024 KiB; 1 GiB = 1024 MiB; 1 TiB = 1024 GiB

1.3 Data storage and compression

Order: bit, nibble, byte, KiB, MiB, GiB, TiB, PiB, EiB (smallest to largest)

1 nibble = 4 bits; 1 byte = 8 bits = 2 nibbles

1 KiB = 1024 bytes; 1 MiB = 1024 KiB; 1 GiB = 1024 MiB; 1 TiB = 1024 GiB; 1 PiB = 1024 TiB; 1 EiB = 1024 PiB

To a bigger unit divide (by 1024 each step); to a smaller unit multiply

Image file size (bits) = width $\times$ height $\times$ colour depth

Sound file size (bits) = sample rate $\times$ sample resolution $\times$ length in seconds

bits $\div 8$ = bytes; bytes $\div 1024$ = KiB; KiB $\div 1024$ = MiB

Compression = reducing the size of a file

Lossy: data removed permanently (lower resolution, colour depth, sample rate, sample resolution)

Lossless: no data removed permanently; e.g. RLE stores each run as (count, value)

JPEG, MP3 and MP4 are lossy compressed files

2.1 Types and methods of data transmission

Packet = packet header + payload + trailer

Header: destination address, originator's (sender's) address, packet number

Payload: the actual data; trailer: end-of-packet marker and error-checking data

Packet switching: split into packets → each can take a different route → routers control the route → may arrive out of order → reordered once the last packet arrives

Serial: one bit at a time, one wire, bits arrive in order, reliable over long distances

Parallel: several bits at the same time, several wires, faster, short distances only (skew, crosstalk)

Simplex: one direction only

Half-duplex: both directions, but only one direction at a time

Full-duplex: both directions at the same time

USB (universal serial bus): serial, one bit at a time down a single wire

2.2 Methods of error detection

Errors in transmission: interference or crosstalk, causing data loss, data gain or data change

Even parity: even number of 1s in each byte; odd parity: odd number of 1s

Parity bit: set to 0 or 1 to make the count of 1s match the agreed parity

Parity misses an even number of changed bits and transposed bits

Parity block: the error is where the failing byte (row) and failing bit (column) cross

Checksum: calculated from the data with an algorithm, sent with it, recalculated and compared

Echo check: a copy is sent back and compared with the original by the sender

Check digit: calculated from the other digits and added to the number; recalculated on data entry (ISBN, barcode)

ARQ: acknowledgement (positive or negative) and timeout; resend until acknowledged

2.3 Encryption

Purpose of encryption: keep data secure by making it meaningless if intercepted (it doesn't stop interception)

Plain text (before) → encryption algorithm + key → cipher text (after)

Symmetric: one key both encrypts and decrypts; the key is sent to (or generated by) the receiver

Asymmetric: public key encrypts, matching private key decrypts; the public key cannot decrypt

Public key: anyone can request it; private key: never shared, only the owner knows it

Asymmetric is more secure: no key exchange, the private key is never transmitted

3.1 Computer architecture

CPU: processes instructions and data by running the fetch–decode–execute cycle

Microprocessor: a processor on a single integrated-circuit chip

Stored program concept: instructions and data are stored together in memory and fetched and executed one at a time

CU: decodes instructions (using the instruction set), sends control signals to manage the flow of data; contains the CIR

ALU: carries out calculations and logic operations; contains the ACC

PC: address of the next instruction · MAR: address about to be used · MDR: data/instruction to or from memory · CIR: instruction being decoded/executed · ACC: result of a calculation

Address bus: addresses, one way · data bus: data and instructions, both ways · control bus: control signals, both ways

Fetch: PC → MAR (PC + 1) → RAM (address bus) → MDR (data bus) → CIR

Decode: CU decodes the instruction in the CIR using the instruction set (opcode + operand)

Execute: CU sends signals; ALU does the operation; result in the ACC

Clock speed in GHz = billions of FDE cycles per second; cycles per second (all cores) $\approx$ clock speed $\times$ number of cores

More cores: more instructions at the same time · higher clock speed: more per second · larger cache: frequently used data accessed faster

Instruction set: list of all the (machine code) commands a CPU can process

Embedded system: dedicated function, dedicated hardware, microprocessor, firmware not easily changed, part of a larger device

3.2 Input and output devices

Input device: captures data from outside and converts it to binary for the computer to process

Output device: converts the computer's binary data into a form useful outside it (image, sound, print, object, movement)

Storage devices (HDD, SSD, DVD, USB flash drive) and memory (RAM, ROM, registers) are neither input nor output

Sensor = input device; actuator = output device; headphones and speakers = output; microphone = input

Sensor: an input device that measures a physical property of its surroundings and sends the data to a computer or microprocessor

Analogue sensor data goes through an ADC before a microprocessor can use it

Control loop: sensor sends data continuously → ADC → microprocessor compares with stored value → outside range: signal to actuator/output; within range: no action → repeat

Resistive: layers pushed together complete a circuit · capacitive: conductive finger changes an electrostatic field · infra-red: touch breaks a grid of beams

The 14 sensors and what each measures (table above)

3.3 Data storage

Primary storage is directly accessed by the CPU; secondary storage is not

RAM: volatile, read/write, holds data and programs currently in use

ROM: non-volatile, read only, holds start-up instructions (bootstrap, BIOS) and firmware

Cache: primary, holds frequently used instructions and data

Secondary storage is non-volatile and stores data, software and the OS permanently

Magnetic: platters, tracks and sectors, read/write head, electromagnets; HDD, magnetic tape

Optical: laser, pits and lands, spiral track; CD, DVD, Blu-ray

Solid-state: NAND/NOR, transistors, control and floating gates, no moving parts; SSD, SD card, USB flash drive

Virtual memory: a partition of secondary storage that pages move to and from when RAM is full

Cloud storage: remote physical servers run by a third party, accessed over the internet

3.4 Network hardware

A device needs a NIC to connect to a network; the NIC sends and receives data and gives the physical connection

MAC address: given by the manufacturer; unique; usually static; identifies the device

MAC structure: 48 bits, 12 hex digits in 6 pairs, colons or hyphens; manufacturer code + serial code

IP address: given by the network; identifies the device so data can be sent to it; static or dynamic

Dynamic IP: a new address each time the device connects, so it can change

IPv4: 32 bits, 4 denary groups 0–255, dots; about 2^{32} addresses

IPv6: 128 bits, 8 hex groups 0000–FFFF, colons; :: replaces zero groups once

Router: forwards data to its destination IP address, assigns IP addresses, connects a network to the internet

Hex digit = 4 bits; two hex digits = 1 byte; denary 0–255 = 8 bits

4.1 Types of software and interrupts

System software provides the services the computer requires: the operating system and utility software

Application software provides the services the user requires, to perform tasks

Utility examples: anti-virus, firewall, defragmentation, backup, compression, disk formatting

Firmware: stored in ROM; includes the BIOS and the bootloader (bootstrap)

From the bottom up: hardware → firmware (bootloader) → operating system → applications, each running on the one before

The 9 OS functions: files, interrupts, interface, peripherals and drivers, memory, multitasking, platform for applications, security, user accounts

Interrupt: a signal to the CPU that something needs its attention

Hardware interrupts (key press, mouse move or click, printer out of paper) and software interrupts (division by zero, two processes using one memory location)

Handling: priority → queue → check after each cycle → save registers on stack → ISR → restore

4.2 Programming languages, translators and IDEs

High-level: English-like statements; easy to read, write and debug; machine independent (portable); must be translated; can't directly manipulate hardware

Low-level: machine code or assembly language; directly manipulates hardware; memory efficient and fast; hard to read, write and debug; not portable

Assembly language uses mnemonics and is translated by an **assembler**

The three translators: compiler and interpreter (for high-level code), assembler (for assembly language)

Compiler: translates everything before anything runs; one report of all errors; produces an executable file

Interpreter: translates and runs one line at a time; halts at an error; carries on after the fix; no executable file

Interpreter during development; compiler for the finished program

IDE tools: code editor, run-time environment, translators, error diagnostics, auto-completion, auto-correction, prettyprint

5.1 The internet and the world wide web

Internet: the infrastructure (networks, cables, routers, devices); WWW: the collection of websites and web pages accessed using the internet

URL: a text-based address for a web page; parts: protocol, domain name, web page/file name

HTTP: protocol for requesting and sending web pages; data unencrypted

HTTPS: HTTP with SSL/TLS; encrypts data; server checked by its digital certificate; padlock

HTTPS steps: browser asks server to identify itself → server sends digital certificate → browser authenticates it; valid: encrypted connection; not valid: rejected, user warned

Web browser: software whose main purpose is to render HTML and display web pages

Browser functions: address bar, bookmarks/favourites, history, tabs, storing cookies, navigation tools

DNS: server holding a database of domain names/URLs and matching IP addresses

Retrieval:

URL → DNS (next DNS if not found) → IP address to browser → request to web server → HTML back → browser renders

Web server: stores web pages; receives requests and sends the requested page

HTML: the language used to write web pages

Cookie: small text file created by a web server, stored and managed by the browser

Session cookie: RAM, temporary, deleted when the browser closes; persistent cookie: secondary storage, permanent until it expires or is deleted

5.2 Digital currency

Digital currency: exists only electronically, never physically

Features: electronic only; decentralised (or centralised); usually encrypted; tracked by a digital ledger

Blockchain: a shared digital ledger of payments; each entry has a time-stamp, and entries can be added but never changed

The process that uses a digital ledger to track digital currency: **blockchain**

Payment steps: encrypt and sign → send to ledger → record with time-stamp in a block → full block gets a hash → added to the blockchain on every device

Each block stores its own hash and the previous block's hash

Records can't be changed or deleted, only added; every transaction can be traced

5.3 Cyber security

Brute force: trial and error, repeated combinations until the password is found, often by software

Data interception: packets captured in transit (packet sniffer) and analysed by a third party

DDoS: malware → bots → botnet → attack started → all bots request at once → server crashes

Hacking: gaining unauthorised access to a computer system

Virus (needs a host, runs when file run); worm (spreads itself over a network); Trojan horse (disguised as genuine software); spyware (records key presses, sends to third party); adware (unwanted adverts); ransomware (encrypts data, demands payment)

Phishing: genuine-looking email → click link → fake website → details stolen

Pharming: malicious code installed → real URL typed → redirected to fake website → details stolen

Social engineering: manipulating people to give up data (phishing, baiting, shoulder surfing)

Access levels: different permissions (none, read, edit, delete) per username

Two-step verification: extra code sent to the user's registered device

Firewall: criteria set, all traffic in and out checked, rejected if it fails; hardware or software

Proxy server: examines requests, limits rate, blocks IPs, caches pages, hides the server's IP address

SSL: browser asks server to identify itself → digital certificate sent → browser authenticates it → encrypted connection; newer version TLS

6.1 Automated systems

Automated system: performs a task without human intervention

Sensor = input (measures, sends data continuously); microprocessor = process (compares with stored values, sends signals); actuator = output (physical movement)

Analogue sensor readings are converted to digital by an ADC before the microprocessor uses them

Loop: sensor sends data → ADC → microprocessor compares with stored value(s) → outside range: signal to actuator; within range: no action → repeat continuously

Timer: start when switched on, reset on each new trigger, switch off when it reaches the limit

Two sensors: act only when both readings meet their conditions

An actuator makes physical movement; screens, speakers and alerts are not actuators. A lamp or heater isn't one, but the relay or switch that turns it on and off is

Advantages: no human intervention, 24/7, faster, more accurate, safer, saves labour costs

Disadvantages: set-up cost, maintenance cost, job losses and deskilling, training, malfunctions, can't handle the unexpected

6.2 Robotics

Robotics: the branch of computer science covering the design, construction and operation of robots

Robot characteristics: mechanical structure (framework); electrical components (sensors, microprocessor, actuators); programmable. It can move

AI is not a characteristic of a robot; some robots use it, many don't

Not a robot: no mechanical structure or no actuators, so it can't move itself

Control loop: sensor sends digitised data continuously → microprocessor compares with stored value → signal to actuator (or no action) → repeat

Areas: industry, transport, agriculture, medicine, domestic, entertainment

Advantages and disadvantages depend on who: owner, workers, or customers/public

6.3 Artificial intelligence

AI: the branch of computer science dealing with the simulation of intelligent behaviour by computers

Main characteristics: collection of data; rules for using that data; ability to reason; can include ability to learn and adapt

Expert system components: knowledge base, rule base, inference engine, interface

Knowledge base: the facts. Rule base: the rules that link the facts. Inference engine: decides the questions and applies the rules to reach the conclusion. Interface: input of data, output of questions and result

Machine learning: a program automatically adapts its own processes and/or data

Machine learning loop: collect data → find patterns → decide or predict → record success or failure → adapt rules or data → repeat

Supervised: trained with inputs and correct outputs. Unsupervised: finds the patterns in inputs by itself

7.1–7.3 Program development and design

Life cycle stages, in order: analysis → design → coding → testing

Analysis: abstraction, decomposition, identifying the problem and the requirements

Design: decomposition, structure diagrams, flowcharts, pseudocode

Coding: writing program code, with iterative testing; testing: running the code with test data

Abstraction: removing details that are not relevant to the problem

Component parts: inputs, processes, outputs, storage

Every system is made of sub-systems, which are made of further sub-systems

Design methods: structure diagram, flowchart, pseudocode

Flowchart symbols: terminator, process, input/output, decision, subroutine, flow line

Purpose = what the algorithm is for; processes = the steps it carries out

7.4 Standard methods of solution

Totalling: $\text{Total} \leftarrow 0$ before the loop; $\text{Total} \leftarrow \text{Total} + \text{Value}$ inside it

Counting: $\text{Count} \leftarrow 0$ before the loop; $\text{Count} \leftarrow \text{Count} + 1$ inside it (usually inside an IF)

Maximum: start with the first value or a very low value; replace when $\text{Value} > \text{Highest}$

Minimum: start with the first value or a very high value; replace when $\text{Value} < \text{Lowest}$

$\text{average} = \frac{\text{total}}{\text{number of values}}$, worked out after the loop

$\text{ROUND}(\text{Value}, \text{Places})$ rounds to that many decimal places

Linear search: check each item in turn from the first; stop when found or past the last item

Bubble sort: compare neighbours, swap with a temporary variable, repeat passes until no swaps

n items: the first pass makes $n - 1$ comparisons; at most $n - 1$ passes

$>$ in the comparison gives ascending order; $<$ gives descending order

7.5 Validation and verification

Validation: the **program** tests each value against **rules** and refuses it if it breaks one

Verification: a check for **slips made while copying or typing**, so the stored value matches its source

Range check: lowest and highest allowed values; a bad value is $X < \text{Low}$ OR $X > \text{High}$

Length check: the number of characters, $\text{LENGTH}(X)$

Type check: the right kind of data, e.g. whole number: $X = \text{ROUND}(X, 0)$

Presence check: the field is not empty, $X <> ""$

Format check: the pattern of characters, $\text{SUBSTRING}(X, \text{Start}, \text{Length})$

Check digit: worked out again from the other digits on entry and compared with the last digit

Visual check: a person reads the screen against the original

Double entry check: keyed twice, the program compares the two copies

WHILE loop: condition for a **bad** value (OR); REPEAT ... UNTIL: condition for a **good** value (AND)

Test data: normal (accepted), abnormal (rejected), extreme (largest/smallest accepted), boundary (accepted and rejected pair at an end)

7.6–7.8 Testing, trace tables and errors

Normal data: inside the allowed limits, should be accepted

Abnormal (erroneous) data: outside the limits or the wrong type, should be rejected

Extreme data: the largest and the smallest acceptable values

Boundary data: the largest or smallest acceptable value **and** the nearest rejected value beyond it

A trace table has a column for each variable, each changing array element and OUTPUT; prompts are outputs

A rogue value ends the loop and is not processed; later inputs are never read

$\text{DIV}(a, b)$ is the whole-number part of $a \div b$; $\text{MOD}(a, b)$ is the remainder: $\text{DIV}(17, 5) = 3$, $\text{MOD}(17, 5) = 2$

$\text{ROUND}(\text{value}, \text{places})$, $\text{LENGTH}(\text{string})$, $\text{SUBSTRING}(\text{string}, \text{start}, \text{length})$ (first character is position 1), UCASE , LCASE

Loop closers: `FOR ... NEXT`, `REPEAT ... UNTIL`, `WHILE ... ENDWHILE`; `IF ... ENDIF`, `CASE ... ENDCASE`

8.1 Programming concepts

`DECLARE Name : TYPE` declares a variable; `CONSTANT Name ← value` declares a constant

Variable: value can change while the program runs. Constant: value is fixed and can't change while it runs

Five data types: `INTEGER`, `REAL`, `CHAR`, `STRING`, `BOOLEAN`

`←` assigns; `=` compares

$\text{DIV}(a, b)$ whole-number quotient; $\text{MOD}(a, b)$ remainder; $^$ power; $/$ ordinary division

Relational: `=` `<>` `<` `<=` `>` `>=`; logical: `AND` `OR` `NOT`

`IF ... THEN ... ELSE ... ENDIF`; `CASE OF ... OTHERWISE ... ENDCASE`

`FOR ... TO ... NEXT` count-controlled; `WHILE ... DO ... ENDWHILE` pre-condition (may run 0 times); `REPEAT ... UNTIL` post-condition (runs at least once)

Totalling: `Total ← Total + Value`; counting: `Count ← Count + 1`; both start at 0 before the loop

$\text{LENGTH}(s)$, $\text{SUBSTRING}(s, \text{start}, \text{length})$, $\text{UCASE}(s)$, $\text{LCASE}(s)$; first character is position 1

Characters from position a to b : $\text{SUBSTRING}(s, a, b - a + 1)$

Comments start with `//`

Maintainable: meaningful identifiers, comments, procedures and functions (also constants, indentation)

8.1 Procedures, functions and library routines

`PROCEDURE Name(P1 : TYPE, ...) ... ENDPROCEDURE`; called with `CALL Name(values)`

`FUNCTION Name(P1 : TYPE, ...) RETURNS TYPE ... RETURN value ... ENDFUNCTION`; called inside an expression, never with `CALL`

Write every procedure and function first, above the main program that uses it; at most 3 parameters each

Values in a call match the parameters by position and must have the right data type

After a procedure, the program carries on from the line after the call; a function's answer is used where its name was written

Global: declared outside subroutines, scope is the whole program. Local: declared inside a subroutine, scope is that subroutine only, removed when it ends

$\text{DIV}(a, b)$ whole-number quotient; $\text{MOD}(a, b)$ remainder; $a = \text{DIV}(a, b) \times b + \text{MOD}(a, b)$

`ROUND(x, places)` rounds to `places` decimal places

`RANDOM()` gives a number from 0 to 1 inclusive; a whole number from Low to High is `ROUND(RANDOM() * (High - Low), 0) + Low`

8.2 Arrays

1D: `DECLARE Name : ARRAY[Lower:Upper] OF TYPE`

2D: `DECLARE Name : ARRAY[Lower1:Upper1, Lower2:Upper2] OF TYPE`; first index row, second column

Number of elements: $Upper - Lower + 1$ in each dimension; a 2D array has rows $\times$ columns

Every element has the same data type; the first index may be 0 or 1

`Name[i]` and `Name[r, c]` are used like variables; the index can be a variable or an expression

An index outside the declared bounds is an error

Fill or process every element with `FOR` over the whole index range; nested `FOR` loops for 2D (outer row, inner column)

Parallel arrays: the same index in each array belongs to the same item

8.3 File handling

`OPENFILE <name> FOR READ` or `OPENFILE <name> FOR WRITE` before a file is used

`READFILE <name>, <variable>` copies the next line of the file into the variable

`WRITEFILE <name>, <variable>` puts the variable's value into the file on a new line

`CLOSEFILE <name>` when the program has finished with the file

Opening for `WRITE` creates the file, or empties it if it already exists

To switch between reading and writing, close the file and reopen it with the other mode

Files keep data on secondary storage after the program ends; variables in RAM do not

`LENGTH(s)` number of characters; `UCASE(s)` capitals; `LCASE(s)` small letters

9 Databases

Field = column; record = row; the heading row is not a record

Primary key: a field whose value is different for every record

Data types: integer, real, Boolean, character, text/alphanumeric, date/time

`SELECT` fields to show `FROM` table `WHERE` condition `ORDER BY` field `ASC/DESC` ;

Comparisons: `=`, `<>` (not equal), `<`, `>`, `<=`, `>=`

`AND`: both conditions true; `OR`: at least one true

`SUM(field)`: total of that field; `COUNT(field)`: number of records

Text values in quotes; numbers and `TRUE/FALSE` without

10 Boolean logic

`NOT`: output is the opposite of the single input

`AND`: 0 unless every input is 1

`OR`: 0 only for the input pair 0, 0

NAND = NOT (AND): 0 for the input pair 1, 1 and 1 for the other three

NOR = NOT (OR): 0 unless neither input is 1

XOR: 1 when exactly one input is 1

The six symbols, as drawn in the syllabus; a circle on the output means "invert"

n inputs give 2^n rows: 2 inputs $\rightarrow$ 4 rows, 3 inputs $\rightarrow$ 8 rows

Row order for three inputs: 000, 001, 010, 011, 100, 101, 110, 111

One gate per operator (each NOT is a gate); non-NOT gates have exactly two inputs
