GCSE

WJEC Eduqas GCSE in

COMPUTER SCIENCE

ACCREDITED BY OFQUAL

SPECIFICATION

Teaching from 2020
For award from 2022

Version 1

This Ofqual regulated qualification is not available for candidates in maintained schools and colleges in Wales.

WJEC Eduqas GCSE (9-1) in COMPUTER SCIENCE

FOR TEACHING FROM 2020
FOR AWARD FROM 2022

GCSE COMPUTER SCIENCE

SUMMARY OF ASSESSMENT

Component 1: Understanding Computer Science
Written examination: 1 hour 45 minutes
50% of the qualification

This component investigates hardware, logical operations, communication, data representation and data types, operating systems, principles of programming, software engineering, program construction, security, authentication and data management and the impacts of digital technology on wider society as well as algorithms and programming constructs.

Component 2: Computer Programming
On-screen examination: 2 hours
50% of the qualification

This component investigates problem solving, programming languages, data structures and data types, program design, implementation and testing. Learners are required to produce a programmed solution to a set task which will then be the basis for examination.

This linear qualification will be available in the summer series each year. It will be awarded for the first time in Summer 2022.

Qualification Accreditation Number: 601/8291/X

GCSE COMPUTER SCIENCE

1 INTRODUCTION

1.1 Aims and objectives

The WJEC Eduqas GCSE in Computer Science encourages learners to:

- understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation
- analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs to do so
- think creatively, innovatively, analytically, logically and critically
- understand the components that make up digital systems, and how they communicate with one another and with other systems
- understand the impacts of digital technology to the individual and to wider society
- apply mathematical skills relevant to computer science.

Computers are widely used in all aspects of business, industry, government, education, leisure and the home. In this technological age, a study of computer science, and particularly how computers are used in the solution of a variety of problems, is essential to learners.

Computer science integrates well with subjects across the curriculum. It demands both logical discipline and imaginative creativity in the selection and design of algorithms and the writing, testing and debugging of programs; it relies on an understanding of the rules of language at a fundamental level; it encourages an awareness of the management and organisation of computer systems; it extends learners' horizons beyond the school or college environment in the appreciation of the effects of computer science on society and individuals.

The WJEC Eduqas GCSE in Computer Science has been designed to give an understanding of the fundamental concepts of computer science and a broad scope of study opportunities. This specification has been designed to free centres to concentrate on innovative delivery of the course by having a streamlined, uncomplicated, future-proof structure, with realistic technological requirements.

1.2 Prior learning and progression

Although there is no specific requirement for prior learning, this specification builds on the knowledge, understanding and skills established through the computer science elements of the computing programme of study at Key Stage 3 and enables learners to progress into further learning and/or employment.

This specification provides a suitable foundation for the study of computer science at AS and A level. In addition, the specification provides a coherent, satisfying and worthwhile course of study for learners who do not progress to further study in this subject.

In addition this specification is not age specific and, as such, provides opportunities for learners to extend their life-long learning.

1.3 Equality and fair access

This specification may be followed by any learner, irrespective of gender, ethnic, religious or cultural background. It has been designed to avoid, where possible, features that could, without justification, make it more difficult for a learner to achieve because they have a particular protected characteristic.

The protected characteristics under the Equality Act 2010 are age, disability, gender reassignment, pregnancy and maternity, race, religion or belief, sex and sexual orientation.

The specification has been discussed with groups who represent the interests of a diverse range of learners, and the specification will be kept under review.

Reasonable adjustments are made for certain learners in order to enable them to access the assessments (e.g. candidates are allowed access to a Sign Language Interpreter, using British Sign Language). Information on reasonable adjustments is found in the following document from the Joint Council for Qualifications (JCQ): *Access Arrangements and Reasonable Adjustments: General and Vocational Qualifications.*

This document is available on the JCQ website (www.jcq.org.uk). As a consequence of provision for reasonable adjustments, very few learners will have a complete barrier to any part of the assessment.

2 SUBJECT CONTENT

This specification promotes the integrated study of computer science. It will enable learners to develop a broad range of skills in the areas of programming, system development, computer architecture, data, communication and applications.

The knowledge, understanding and skills are set out in the two columns in the pages that follow. The topic to be studied is in the first column, with the amplification in the second column. There is no hierarchy implied by the order in which content and amplification are presented, nor should the length of the various sections be taken to imply any view of their relative importance.

The subject content for GCSE Computer Science will be assessed across two components. Whist there is a degree of overlap between the content in Component 1 and Component 2, the context in which this content is assessed differs. In Component 1, content is assessed in a theoretical way, whereas in Component 2 it is assessed through its use within programs.

Component 1: Understanding Computer Science
Written examination: 1 hour 45 minutes
50% of the qualification
100 marks

Component 2: Computer Programming
On-screen examination: 2 hours
50% of the qualification
80 marks

2.1 Component 1

1. Hardware	
Content	**Amplification**
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Central processing unit (CPU)	• the purpose of the CPU • the purpose of different components in the Von Neumann CPU architecture, including: • control unit (CU) • arithmetic and logic unit (ALU) • main memory • cache • registers • program counter (PC) • current instruction register (CIR) • accumulator (ACC) • memory address register (MAR) • memory data register (MDR) • input and output devices.

Content	**Amplification**
	• the role of the components of the CPU in the fetch-decode-execute cycle • CPU performance in relation to: • cache size and levels of cache • clock speed • number of cores.
Primary storage	the use and characteristics of primary storage, including: • random access memory (RAM) • read only memory (ROM) • flash memory • cache memory • virtual memory.
Additional hardware components	the role of additional hardware, including: • integrated and dedicated GPUs • sound cards • motherboards.
Secondary storage	• the functional characteristics of contemporary secondary storage technologies, including: • optical • magnetic • solid state • cloud storage. • the suitability and characteristics of contemporary secondary storage media in terms of: • capacity • durability • portability • speed • cost.
Embedded systems	• the characteristics and purpose of embedded systems • example uses of embedded systems.

2. Logical operations	
Content	**Amplification**
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Logical operators	the use of logical operators in truth tables and to solve problems, including: • `AND` • `OR` • `NOT` • `XOR.`
Boolean logic	The simplification of Boolean expressions using Boolean identities and rules *(see Appendix B)*.

3. Networking and cybersecurity	
Content	**Amplification**
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Networks	• the characteristics of networks, and their advantages and disadvantages • common network topologies, including star and mesh, and their advantages and disadvantages over ring and bus topologies • the hardware required to establish wired and wireless connectivity, including: • routers • hubs • switches • bridges • wireless access points (WAPs) • network interface cards (NICs) • the importance of creating networking standards • the purpose of each layer in the 7-layer Open Systems Interconnection model (OSI model) • the use of contemporary networking protocols in the 7-layer OSI model, including: • HTTP • HTTPS • SMTP • FTP • TCP • IP • Ethernet • WiFi (802.11) • the typical contents of a TCP/IP packet and packet switching • methods of routing traffic on a network and calculation of routing costs.

Content	Amplification
Internet	• the purpose of Domain Name System (DNS) servers and how they work • the structure of URLs and the role and function of a web browser.
Cybersecurity	• the characteristics of different threats to computer systems, including: • malware • phishing • social engineering • brute force attacks • denial of service attacks • data interception and theft • SQL injection • different ways of protecting against threats during system design, creation, testing and use, including: • penetration testing • network forensics • anti-malware software • firewalls • user access levels • passwords • double authentication • encryption.

4. Data representation and storage	
Content	**Amplification**
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Data types	data types, including: • integer • real • character • string • Boolean.
Representation of numbers	• use and convert between denary, binary and hexadecimal counting systems • the representation of positive and negative integers in a fixed-length store using both two's complement, and sign and magnitude representation • binary addition and subtraction techniques • the concept of overflow and underflow • arithmetic shift functions and their effect.

Content	Amplification
Representation of characters	• the digital storage of characters • standardised character sets, including: • Unicode • American Standard Code for Information Interchange (ASCII).
Representation of graphics	• the digital storage of graphics • the effect of different resolutions and colour depths on the quality of graphics • how to calculate the storage requirements for graphics using different dimensions and colour depths.
Representation of sound	• the digital storage of sound • the effect of different sampling rates on the quality of sound • how to calculate the storage requirements for sound using different sampling rates.
Storage requirements	• the relationship between data storage units using the Base 2 prefixes, including: • bit • nibble • byte • kilobyte (i.e. 1,024 bytes) • megabyte (i.e. 1,024 kilobytes) • gigabyte (i.e. 1,024 megabytes) • terabyte (i.e. 1,024 gigabytes) • petabyte (i.e. 1,024 terabytes) • the calculation of data capacity requirements.
Compression	• the principles of data compression and the use of different compression types, including: • lossy • lossless • the calculation of compression ratios.

5. Data organisation

Content	Amplification
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Data structures	• the design, interpretation and manipulation of data structures including: • records • one-dimensional arrays • two-dimensional arrays • the selection, identification and justification of appropriate data structures for different situations.
File design	The design of files and records appropriate for a particular application.

6. Operating systems

Content	Amplification
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Operating systems	The purpose of the operating system.
	how the operating system manages: • CPU • multi-tasking • interrupts • memory • backing store • peripherals • interface • security.
Utility software	the purpose and functionality of a range of utility software, including: • anti-virus • clipboard manager • system profiles • backup software • disk checkers • disk compression • disk defragmenters • disk formatters • disk partition editors • archivers • cryptographic utilities • data recovery • revision control • file managers.

7. Principles of programming

Content	Amplification
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Levels of computer language	• the characteristics and purpose of high-level and low-level languages • situations that require the use of a high-level or a low-level language.

8. Algorithms and constructs	
Content	**Amplification**
	Candidates should be able to demonstrate and apply knowledge and understanding and analyse problems in computational terms on the following:
Algorithms	• methods of defining algorithms, including pseudo-code and flowcharts *(see Appendix A)* • writing, correcting, testing and interpreting the function of algorithms that solve problems using: • subroutines • input • output • sequence, selection and iteration • counts and rogue values • local and global variables • self-documenting identifiers • string handling • mathematical and logical operations *(see Appendix A).*
Sorting	the characteristics of sort algorithms, including: • merge sort • bubble sort.
Searching	the characteristics of search algorithms, including: • linear search • binary search.
Validation and verification	• the purpose of and techniques for data validation and verification • algorithms that validate and verify data.

9. Software development	
Content	**Amplification**
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Software tools	Integrated Development Environment (IDE) tools, including: • editor • automatic formatting • automatic line numbering • automatic colour coding • linker • libraries • routines/subroutines • standard functions • loader • debugger • syntax error detection • error diagnostics • trace • break point • variable watch • memory inspector • statement completion • code optimisation • compilation and interpretation of code.

10. Program construction	
Content	**Amplification**
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Translators	• the purpose of the three common types of program translator: • compilers • interpreters • assemblers • the appropriate use of the three common types of program translator • the principal stages involved in the compilation process: • lexical analysis • syntax analysis • semantic analysis • code generation • code optimisation • examples of programming errors: • logical • syntactical • run-time/execution • linking • rounding • truncation

11. Impacts of digital technology on wider society

Content	Amplification
	Candidates should be able to demonstrate and apply knowledge and understanding on the following:
Impacts	The ethical, legal, cultural, environmental and privacy issues linked to the use of computer systems.
Legislation	the impact of relevant current legislation on Computer Science, including: • the General Data Protection Regulation (GDPR) and Data Protection Act 2018 • Computer Misuse Act 1990 • Copyright Designs and Patents Act 1988 • Creative Commons Licensing • Regulation of Investigatory Powers Act 2000 • Telecommunications Regulations Act 2000 • Freedom of Information Act 2000.
Professional standards	The importance of conforming to professional standards, including formal and informal codes of ethical behaviour and conduct.

2.2 Component 2

Structure of assessment

This unit is assessed entirely by an on-screen examination. Candidates will require access to Python 3 and a word processor. The version of Python 3 required will be specified on the scenario issued to candidates and on the examination paper.

Before the on-screen examination

Candidates will be provided with a scenario containing a list of requirements which will be issued to centres on 01 September in the year prior to the award. This scenario will provide information about all tasks to be undertaken by candidates, both individually and in groups, in preparation for the on-screen examination.

During the on-screen examination

Candidates will be issued with a question paper, a prototype Python 3 file and any supporting files. The question paper will contain some additional requirements. These additional requirements will need to be completed individually by each candidate and will require them to use, evaluate and make changes to the Python 3 program provided in the examination.

Additional examination questions will be answered in a word-processed document.

See section 3.2 below for rules regarding the administration of preparatory work and the on-screen examination.

Component content:

Systems Analysis	
Content	**Amplification**
	Working individually and with others the candidates should be able to apply knowledge and understanding and analyse problems in computational terms in the following:
Investigation	• use a systematic approach to problem solving including the use of decomposition and abstraction • use abstraction effectively to model selected aspects of the external world in an algorithm or program • use abstraction effectively to appropriately structure programs into modular parts with clear, well-documented interfaces • analyse a set of requirements • program a solution that meets a set of requirements.

Content	**Amplification**
	Working individually and with others the candidates should be able to apply knowledge and understanding and analyse problems in computational terms in the following:
Design	• design and document the input and output facilities required to produce an effective user interface • design and document all required data structures • using pseudo code, design and document the following routines: • validation and verification • data handling and processing • authentication.
Implementation	design, write, test and refine Python 3 code using the following skills: • create new and extend existing functions or methods • create new and edit existing objects • create new and extend existing Python 3 libraries • use variables (labels), operators, inputs, outputs and assignment • use a variety of data types, including integer, Boolean, real, character and string • use programming constructs to control the flow of a program, including: • iteration (condition and counter controlled loops) • selection • sequence • use basic file handling, including: • open a file • read from a file into a variable • read from a file into an array • write to a file • write to a file from an array • close a file • create new and extend data structures and fixed length records to store data • use string manipulation • create new and extend lists, tuples and dictionaries (arrays) • use slicing • use mathematical and logical operations • create new and modify existing intuitive graphical user interfaces using the Python 3 built in libraries including: • text boxes • buttons • forms

Content	Amplification
	• create and modify data validation and verification routines • create and modify authentication management routines • create code for the solution that is self-documenting and uses meaningful identifiers • use a programming style that is consistent, including indentation and appropriate use of white space • use subroutines with well-defined interfaces • annotate code so that it is accessible to a competent third party • explain the solution or changes made to a solution.
Testing	• design and document an effective testing strategy that will ensure that the final solution meets the requirements • describe how the outcomes of the testing process can be used to inform further development of the solution • design test data to include examples of typical, extreme and erroneous content • implement a test plan using typical, extreme and erroneous data where appropriate • present test outcomes with detailed and informed commentaries • demonstrate testing and refinement of code during development or in response to change in the requirements provided • explain the outcome of testing using given data or data that the candidate has designed.
Refinement	• demonstrate understanding of the requirements and any changes required by revised requirements • demonstrate understanding of changes by presenting evidence of developmental changes • demonstrate testing and refinement of code to improve efficiency • discuss the strengths and weaknesses of differing approaches to meeting the requirements • demonstrate an understanding of the technical terminology/concepts used during software development • evaluate the outcomes of the original given code • evaluate the outcomes of the changes made by additional requirements.

3 ASSESSMENT

3.1 Assessment objectives and weightings

Below are the assessment objectives for this specification. Learners must:

AO1
Demonstrate knowledge and understanding of the key concepts and principles of computer science

AO2
Apply knowledge and understanding of key concepts and principles of computer science

AO3
Analyse problems in computational terms:

- to make reasoned judgements
- to design, program, evaluate and refine solutions.

The table below shows the weighting of each assessment objective for Component 1 and Component 2 and for the qualification as a whole.

	AO1	AO2	AO3
Component 1	30%	15%	5%
Component 2	0%	25%	25%
Total	**30%**	**40%**	**30%**

3.2 Arrangements for Component 2

Some tasks in Component 2 will require work to be completed using Python 3, which is an integrated development environment (IDE) freely available for legal download.

WJEC Eduqas will only support systems written using the version of Python 3 specified on the cover of the issued task requirements. Only the standard Python 3 download should be used with no additional libraries installed.

Before the on-screen examination

This component requires candidates to undertake a substantial piece of work, over an extended period of time, in preparation for the on-screen examination. The sample assessment material provides an example of this.

Although this preparatory work does not form part of the assessment for this component, it will give candidates an essential platform for the work that will be undertaken and assessed in the on-screen examination.

Before the on-screen examination, candidates may use any resource available to them.

There is no requirement for centres to mark this preparatory work.

The requirements will be set annually by WJEC Eduqas and published on the WJEC secure website. Each set of requirements will have a shelf life of one academic year and will only be accepted for submission in the academic year after its publication. A new set of requirements will be released each series from 01 September 2021 (for assessment from 2022).

Centre preparation in readiness for the on-screen examination

Centres will receive media files from WJEC Eduqas in advance of the examination. These will include a solution to the released task for candidates to work on in the examination.

Candidates must not have access to their own materials during the examination.

During the on-screen examination

This assessment will be carried out in accordance with the instructions set out in ‘Instructions for conducting on-screen tests’, Appendix 1 of *Instructions for conducting examinations* (Joint Council for Qualifications). This document is available on the JCQ website (www.jcq.org.uk).

Candidates will need access to a computer with:

- a ‘clean’ user area or storage device on which to save their work
- no access to the Internet, email or other public/network drives
- access to a word processor or similar software to produce their responses
- a functional copy of the appropriate version of Python 3 (referred to on the task cover sheet) with no additional libraries added
- access to the media files supplied by WJEC Eduqas

This practical assessment should be carried out under formal supervision, i.e. the candidates must be in direct sight of the supervisor at all times. Use of resources is tightly prescribed and interaction with other candidates is forbidden.

Submission of assessments

Candidate work for Component 2 must be submitted to WJEC electronically, using the *Surpass* system.

The submission should include all work contained in the candidate's examination folder.

The files expected are:
1. A word processed document with the candidate's responses.
2. The Python 3 file(s) with the changes required by the examination implemented by the candidate.
3. Any data files created by the Python 3 solution.

This work must be submitted as a single zip file.

The naming convention of this zip must be: The centre number, candidate number and the first two initials of the candidate's surname and the first initial of the candidate's first name. For example, Diane Smith, Centre number 12999, candidate number 12345 would store her work in a folder named 12999_12345_SM_D

The candidate may wish to use sub folders to organise their work for each section, they must ensure that the folders are organised in such a way that will allow the WJEC to easily find the relevant file. All of the above must be within the single zip file for submission.

The final upload of the zip file does not form part of the examination and may be carried out by the examination officer together with any technical support assistance as may be required after the examination has ended.

The WJEC will allow a finite window following the examination to upload the work.

Centres are reminded to check the WJEC circulars for any further instructions on this aspect.

4 TECHNICAL INFORMATION

4.1 Making entries

This is a linear qualification in which all assessments must be taken at the end of the course. Assessment opportunities will be available in the summer series each year, until the end of the life of this specification. Summer 2022 will be the first assessment opportunity.

A qualification may be taken more than once. Candidates must resit all examination components in the same series.

The entry code appears below.

WJEC Eduqas GCSE Computer Science: C500QS

The current edition of our *Entry Procedures and Coding Information* gives up-to-date entry procedures.

Centres entering candidates for WJEC Eduqas GCSE (9-1) Computer Science will be required to provide a practical programming statement. This statement confirms that the centre has taken reasonable steps to ensure that each candidate entered for WJEC Eduqas GCSE (9-1) Computer Science has had the opportunity to undertake a programming task or tasks during their course of study which allow(s) them to develop the skills to:

- design
- write
- test
- refine programs

using one or more high-level programming language with a textual program definition, either to a specification or to solve a problem (or problems). In relation to the WJEC Eduqas GCSE (9-1) Computer Science specification this means, as a minimum, using Python 3 to solve the problem issued by WJEC for the year of assessment.

4.2 Grading, awarding and reporting

GCSE qualifications are reported as a grade on the scale from 9 to 1, where 9 is the highest grade. Results not attaining the minimum standard for the award will be reported as U (unclassified).

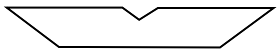
APPENDIX A

Conventions followed in specification

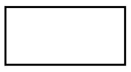
Note 1

Where Von Neumann architecture is represented diagrammatically, the following symbols are used:

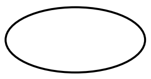
Arithmetic logic unit



Register



Control unit



Note 2

Where candidates are required to apply computing-related mathematics, the following arithmetical and relational operators will be used:

Operator	Meaning	Example
>	Greater than	A>B will return TRUE if the value of A is higher than the value of B otherwise it will return FALSE.
<	Less than	A<B will return TRUE if the value of A is lower that the value of B otherwise it will return FALSE.
<=	Less than or equal to	A<=B will return TRUE if A is the same as or lower than B otherwise it will return FALSE.
>=	Greater than or equal to	A>=B will return TRUE if A is the same as or higher than B otherwise it will return FALSE.
<>	Not equal to	A<>B will return TRUE if A is not the same as B but FALSE if A is the same as B.
EQUALS (usually =)	The same as	A=B will return TRUE if A is the same as B otherwise it will return FALSE.
AND	Both statements must be true for the argument as a whole to be true.	(A=1) AND (B=4) will return TRUE if A is 1 and B is 4. It would return FALSE in all other situations.
OR	Only one of the statements needs to be true for the argument as a whole to be true.	(A=1) OR (B=4) will return TRUE if A is 1 or B is 4. It would only return FALSE if A is not 1 and B is not 4.
NOT	The opposite of	NOT(A) will return TRUE if A is FALSE and FALSE if A is TRUE.

XOR	The argument is false if both statements are true. The argument is false if both statements are false. Otherwise the statement is true.	A XOR B would return TRUE if A and B are different values.
DIV	Integer division Finds the quotient or the 'whole number of times' a divisor can be divided into a number.	11 DIV 2 = 5 The quotient is 5 as 2 divides into 11 a whole number of 5 times
MOD	Modulo division Finds the remainder when a divisor is divided into a number.	11 MOD 2 = 1 The remainder is 1 as 2 divides 5 times into 11 with '1 remaining'

Note 3

Algorithms written in pseudo code will be represented using the following convention:

Construct	Example usage

Declare subroutines	`Declare CapitalLetterOfName`
	`End Subroutine`

Call a subroutine	`call SubroutineNeeded`
Declare and use arrays	`myarray[99]`
Literal outputs	`output "Please enter a number"`
Variable names	`myvariable`
Define variable data type	`myvariable is integer`

Data types	`integer, character, string,`
	`boolean, real`

Assignment	`set counter = 0`
Selection	`if . . . else . . . end if`

Indent at least single space after if or do or repeat etc.	`if counter = 1`
	` output counter`
	`end if`

Annotation	`{Some annotation goes here}`
Comments	`/** Comments for program */`

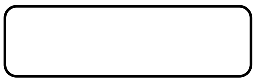
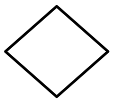
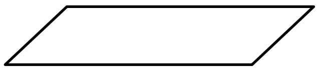
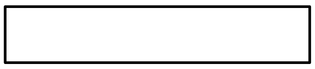
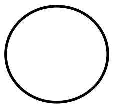
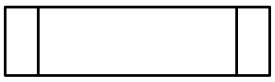
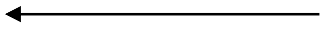
Repetition	`for i . . . next i`
	`repeat . . . until`
	`do . . . loop`
	`do . . . while`
	`while . . . repeat`

String handling	`mid(string,x,y)`
	`left(string,x)`
	`right(string,x)`
	`instring(x,stringa,stringb)`
	`len(string)`
	`val(string)`
	`int(string)`
	`trim(string)`
	`char(number)`

Logical operators `AND, OR, NOT and XOR` will be in upper case.
Logical `TRUE` and `FALSE` will be in upper case.

Note 4

Algorithms represented using a flowchart will use the following convention:

Start / Stop procedure	
Decision box	
Input / Output	
Operation	
Connector	
Store / Subroutine call	
Flow of control (Arrowhead indicates direction of flow)	

Appendix B

Boolean Algebra Conventions

Condition	Symbol	Example
AND	.	$A.B$
OR	+	$A + B$
NOT	overline	$\overline{A}$
XOR	$\oplus$	$A \oplus B$

Boolean Algebra Rules

Rule	Boolean expression
Annulment	$A + 1 = 1$
	$A.0 = 0$
Identity	$A + 0 = A$
	$A.1 = A$
Commutative	$A + B = B + A$
	$A.B = B.A$
Complement	$A + \overline{A} = 1$
	$A.\overline{A} = 0$
Idempotent	$A + A = A$
	$A.A = A$
Distribution	$A.(B + C) = A.B + A.C$
	$A + (B.C) = (A + B).(A + C)$
Absorption	$A + (A.B) = A$
	$A.(A + B) = A$
Association	$A + (B + C) = (A + B) + C = A + B + C$
	$A.(B.C) = A.B.C$