



GCSE Computer Science

How to Revise....

Assessment of course – lots of changes including January JCQ inspection outcome

Old Assessment (2012 - 2017)

Computer Systems (01): 80 marks, 1 hour and 30 minutes, written paper. 40% of total GCSE.

Computational thinking, algorithms & programming (02): 80 marks, 1 hour and 30 minutes, written paper. 40% of total GCSE.

Programming project (03/04): 40 marks, totalling 20 hours, non-exam assessment. 20% of total GCSE.

New Assessment (2018 - 20)

Computer Systems (01): 80 marks, 1 hour and 30 minutes, written paper. 50% of total GCSE.

Computational thinking, algorithms & programming (02): 80 marks, 1 hour and 30 minutes, written paper. 50% of total GCSE.

Programming project (03/04): Consultations are ongoing for the arrangements for the NEA (non-examinable assessment). Currently, all students are required to complete a programming task for a duration of 20 hours. It is not part of the qualification assessment and is therefore not awarded any marks.



Component 1 - Computer Systems

- 1.1 Systems Architecture, Memory and storage
- 1.4 Networks, topologies and layers
- 1.6 System Security
- 1.7 System Software
- 1.8 Ethical, legal, cultural and environmental concerns

Structure of Assessment

Computer Systems (01): 80 marks, 1 hour and 30 minutes, written paper. 50% of total GCSE.

Computational thinking, algorithms & programming (02): 80 marks, 1 hour and 30 minutes, written paper. 50% of total GCSE.



Component 2 - Computational Thinking, Algorithms and Programming

- 1.1 Algorithms
- 2.2 Logic and Languages
- 2.5 Translators and facilities of languages
- 2.6 Data Representation

Quizzes for all units

- Component 1
- Component 2



Useful Resources:

Computing Checklist.xlsx

BBC Bitesize - GCSE Computer Science
- www.bbc.com/education/subjects/z34k7ty

Computer Science crash course
- www.youtube.com/watch?v=tpIctyqH29Q&list=PL8dPuuaLjXtNIUrzyH5r6jN9uIlgZBpdo

Edit P

Firefly > Resources > Subjects > Computing

Under Component Headings.

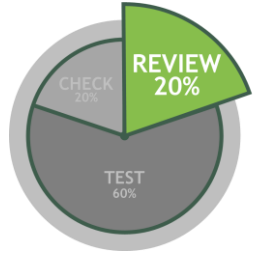
All class presentations can be found under each component.

Each topic consists of key objectives and words as well as diagrams and illustrations.

Quizzes – Exam type questions with answers.

Useful resources

Websites and videos relevant to specification.



Personal learning checklist (on Firefly)

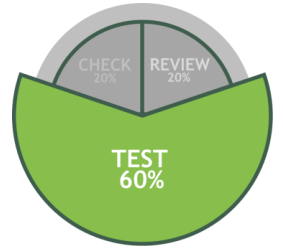
GCSE Computing (J276): Personal Progress Record Component 01: Computer systems

| 1.1 Systems Architecture | | | |
|--|-----|-------|-------|
| | RED | AMBER | GREEN |
| 1. The purpose of the CPU | | | |
| 2. Von Neumann architecture | | | |
| <i>MAR (Memory Address Register)</i> | | | |
| <i>MDR (Memory Data Register)</i> | | | |
| <i>Program Counter</i> | | | |
| <i>Accumulator</i> | | | |
| 3. Common CPU components and their function: | | | |
| <i>ALU (Arithmetic and Logic Unit)</i> | | | |
| <i>CU (Control Unit)</i> | | | |
| <i>Cache</i> | | | |
| 4. The function on the CPU as fetch and execute instructions stored in memory | | | |
| 5. How common characteristics of CPUs affect their performance: | | | |
| <i>Clock speed</i> | | | |
| <i>Cache size</i> | | | |
| <i>Number of cores</i> | | | |
| 6. Embedded systems: | | | |
| <i>Purpose of embedded systems</i> | | | |
| <i>Examples of embedded systems.</i> | | | |

Personal checklist has been created directly from the specification to support students to focus on key objectives for that topic.

Revision should be based on understanding the key term and being able to explain the processes.

Sample Papers created by OCR and old spec past papers are also available on the Firefly KS4 Computing page.



Past paper exam techniques

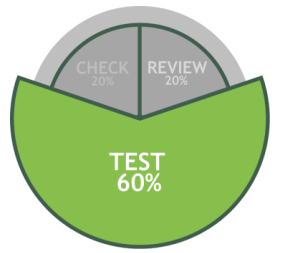


– key to answering the question is understanding the command word of the question

i.e.

- Describe
- Explain
- Discuss
- Identify

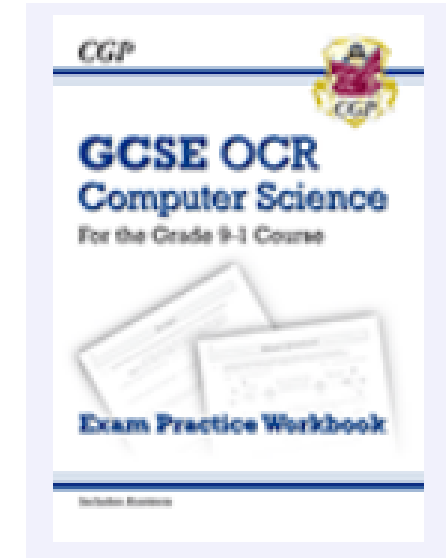
To focus on what is required we use the check methods in the circles.

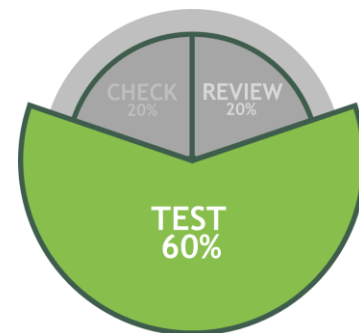


Using the CGP Revision books

All students have purchased the CGP revision guides and exam practice workbooks.

Answers are in the back of the workbooks so students are able to check their answers and if necessary improve by adding detail.





Revision Activity

Section Six — Design, Testing and IDEs 57

Defensive Design DEFINITION

On p22 you saw that insecure databases can be a security threat — unfortunately, every program that interacts with a user can be a risk. To keep programs safe from tampering you need to use defensive design.

Defensive Design helps to ensure programs Function Properly

1) When programs are functioning correctly they should never break and never produce errors. In practice this is difficult to achieve — even the biggest software companies need to update and patch their programs regularly.

2) Programmers try to protect their programs through defensive design — they will try to:

- Anticipate how users might misuse their program, then attempt to prevent it from happening.
- Ensure their code is well-maintained (see p30).
- Reduce the number of errors in the code through testing (see p59-60).

Make sure the Inputs can't be Exploited

1) The easiest way for a user to accidentally or intentionally misuse a program is when entering data.

2) There are two ways that you can prevent users from entering something you don't want them to:

INPUT SANITISATION — removing any unwanted characters before passing data through the program.

INPUT VALIDATION — checking if data meets certain criteria before passing it into the program. (e.g. checking that an email address contains an @ symbol and has a suitable ending (.com, .co.uk, etc).

DEFINITION
A whitelist is a list of all data that a program should accept — any data that isn't on the whitelist will be rejected.

DEFINITION
A blacklist is a list of all data that a program should reject — any data that isn't on the blacklist will be accepted.

TERM: DEFENSIVE DESIGN
DEF: TO KEEP PROGRAMS SAFE FROM TAMPERING. HELPS TO ENSURE PROGRAMS FUNCTION PROPERLY.

DEFINITION
A whitelist is a list of all data that a program should accept — any data that isn't on the whitelist will be rejected.

DEFINITION
A blacklist is a list of all data that a program should reject — any data that isn't on the blacklist will be accepted.

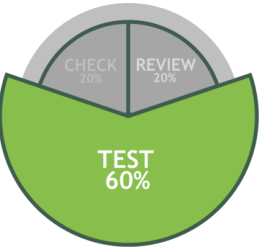
```
FUNCTION FormatName(FILE AS STRING) AS STRING
VAR s AS INT = 0
WHILE s = FILE.Length
SWITCH FILE(s)
CASE " ":
FILE = FILE.RemoveChar(s)
CASE "-":
FILE = FILE.RemoveChar(s)
CASE default:
FILE = FILE.RemoveChar(s)
s = s + 1
ENDSWITCH
ENWHILE
RETURN FILE
ENDFUNCTION
```

I'm taking my wife hiking at the weekend — it's our first validate...

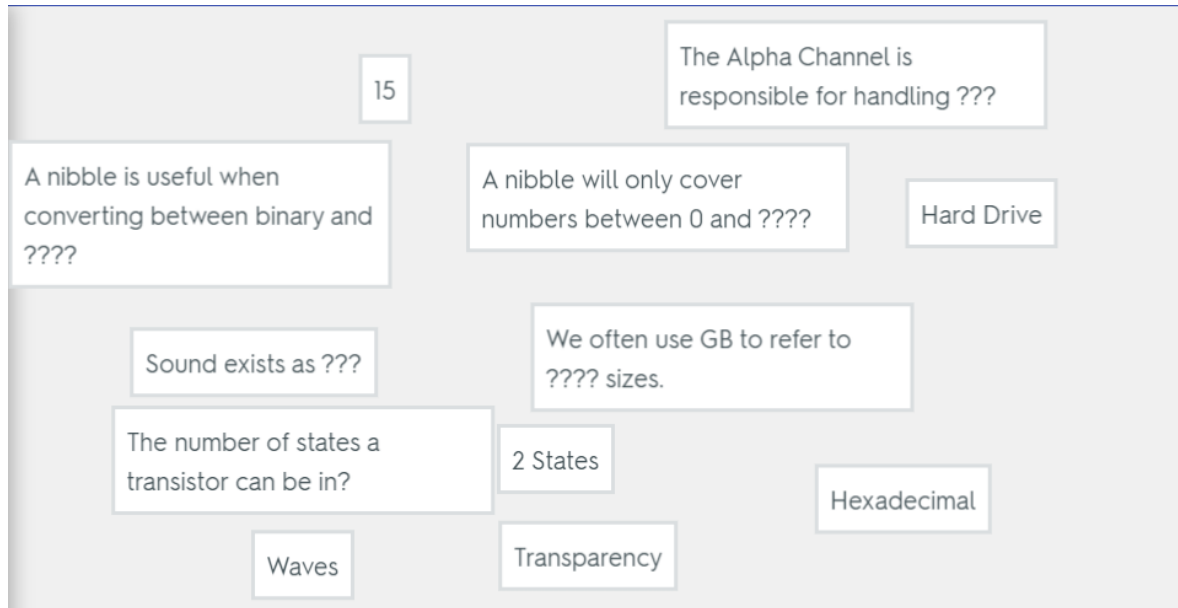
When you're writing programs you should think about what you do and don't want a user to enter, but you can put too many restrictions on the data inputted to your program. When validation and sanitisation start to affect the functionality or impact the user's experience then you've got too much defensive design.

Section Six — Design, Testing and IDEs

1. Use revision guide to define key terms for Defensive Design p.57
2. Using online software create flash cards for key terms.
3. Log on to quizlet.com and join class <https://quizlet.com/join/P5FWEgywU>
4. Create flash cards for key terms.
5. Test knowledge using test mode or match
6. There are already 16 sets of cards for you to use.



Example quizlet match activity



I have copied 16 sets of flash cards to the class for students to use to support most topics.

Students can use pre-existing flash cards to test their knowledge.