GCSE Computer Science
Topic 2.3 Robust Programs

Why defensive design?
Helps to ensure programs function properly.
- ✓ Not breaking
- ✓ Not producing errors

3 elements of Defensive design:
- Anticipate how users might '**misuse**' their program to prevent it from happening.
- Ensure their code is **well maintained**.
- Reduce the numbers of errors in the code through **testing**.

Planning for contingencies / anticipating misuse
- Computer programs should be designed to COPE with unexpected or erroneous input from users.
- Coders should PLAN for all contingencies that might occur. (accidental and deliberate inputs)

Input validation: Validation checks that data input is sensible, reasonable and appropriate to be processed by the program.

Presence check: Checks that data has actually been entered and the field has not been left blank..

Length check: Checks that a specified number of characters has been entered.

Range check : Checks that the input falls within a certain range. e.g. 1-100

Type check : This checks that the data inputted is a certain data-type e.g. number or letters.

Format check :
Checks that the input is in the correct format e.g.
National insurance number XX999999X

Input sanitisation : Removes any unwanted characters BEFORE passing the data to the program.

Authentication is determining the identity of the user before they can access the program or parts of the program.
This is usually based upon a username and associated password.
TOO MUCH AUTHENTICATION CAN:
- *Affect the functionality of the system.*
- *Can put people off using it.*

Maintainability:
Keeping the code well maintained aids defensive design as it means when editing, improving or testing the code – it is clear and easy to understand what the code should be doing.

Commenting:
#Usually written with // or #
#Comments are useful for explaining what key features of a program do.
#Well written/clear comments are essential in allowing other programmers to understand your program.

Indentation :
This is used to separate different statements in a program. This allows other programmers to see the flow of a program more clearly and pick out the different features.
Indentation is usually used to show which statements are part of a previous line of code.
E.g. with ***selection*** and ***iteration***.

Tab

Test Plan	A test plan will outline exactly what you're going to test and how you are going to test it. It should cover all the possible paths through a program.
Normal data	Data that the user is LIKELY to input into the program. Data that the program should be able to process.
Extreme / Boundary data	Values at the limit of what the program should be able to handle. This data should still be able to be processed by the program.
Erroneous data	Data that the program should not accept; usually the wrong data type.

Naming Variables:
Variables should be named so that they reflect their purpose.
This helps other programmers keep track and recognise what the variables are when reading /using the program.

- **Testing** ensures that the software produces the expected results and meets the needs of the user.
- Testing makes sure the program is robust.
- Testing should be destructive and should try to find errors rather than just proving the program works.

ITERATIVE TESTING: Tests carried out whilst the program is being developed. The test results are then used to guide further improvements.

FINAL TESTING: This is carried out once the software has been developed.
Alpha testing is done by the developers.
Beta testing is carried out by the potential users of the software.

A **syntax error** occurs when the compiler or interpreter doesn't understand something the user has typed because it doesn't follow the rules or grammar of the programming language. Syntax errors produce a error message which details what is wrong and which line of code contains the error.

Logical errors: The interpreter / compiler will be table to run the code, but the program will do something unexpected.
E.g. using the wrong Boolean operator.
Logical errors are difficult to diagnose / track down.
Logical errors can only be found through testing, using a test plan.

GCSE Computer Science - Topic 2.3 Robust Programs

What I need to know:

Why do programmers aim for defensive design within their programs?			
What are the 3 elements of defensive design?			
What does planning for contingencies/anticipating misuse entail?			
What is input validation?			
What does a presence check test?			
What does a length check test?			
What does a range check test?			
What does a type check test?			
What does a format check test?			
What is input sanitisation?			
What is authentication?			
What is maintainability?			
How does commenting help improve maintainability?			
How does indentation help improve maintainability?			
How does naming variables help improve maintainability?			
Why are programs tested?			
What is iterative testing?			
What is final testing?			
What is a syntax error?			
What is a logical error?			
What is a test plan?			
What are the three types of data a program should be tested with?			
Define normal, extreme and erroneous data.			