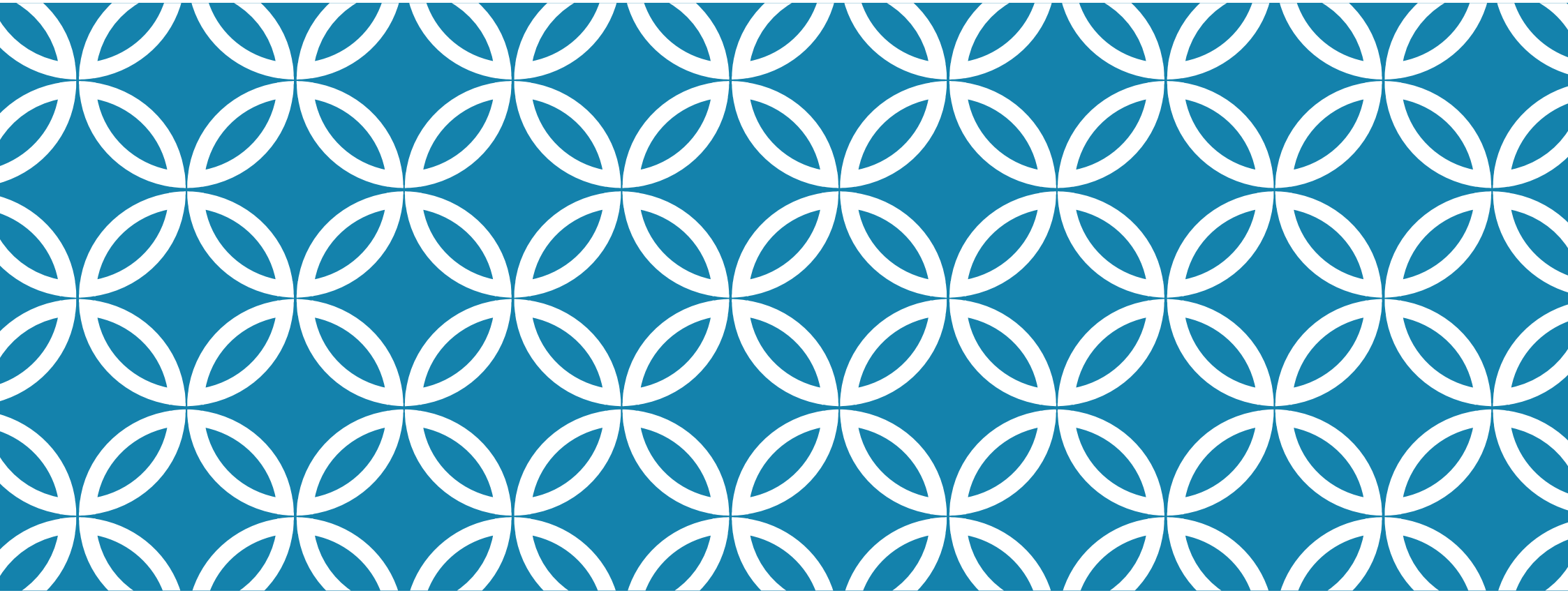


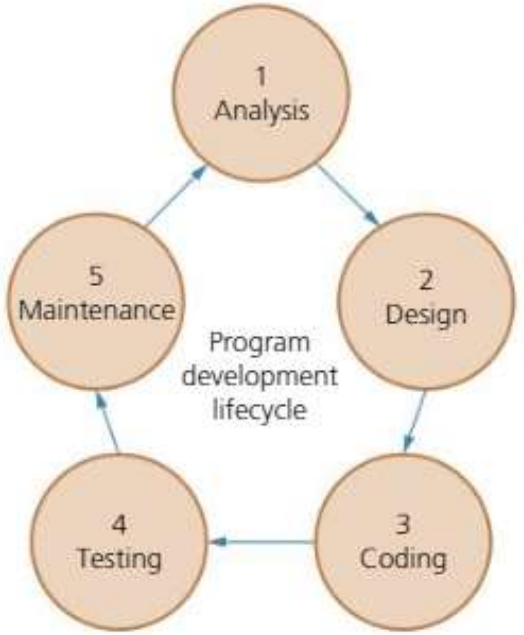
PROGRAM DEVELOPMENT LIFE CYCLE

Chapter 10

THE PROGRAM DEVELOPMENT LIFE CYCLE

- Analysis
  - Understanding business/problem
  - Identifying requirements and problems
  - Tools: feasibility study, questionnaire etc.

- Design
  - How the program should be developed
  - What is to be done
  - User interface, data structure
  - Tools: structure chart, flowcharts and pseudocode



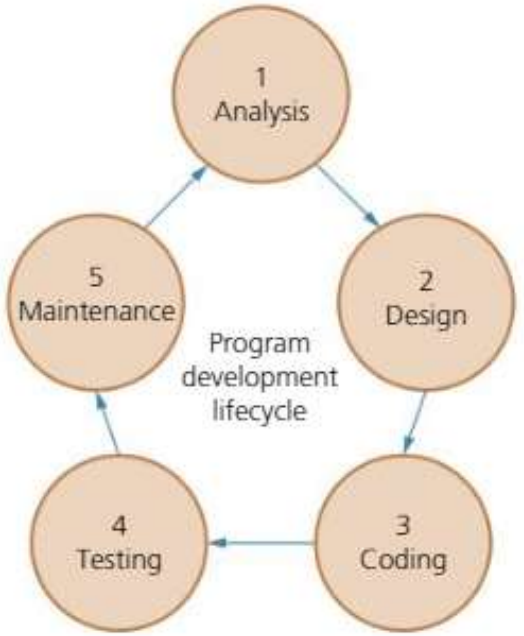
▲ **Figure 12.1** The program development lifecyle

THE PROGRAM DEVELOPMENT LIFE CYCLE

- Coding and iterative testing
  - Writing programing language and then get it tested
  - Iterative testing means that modular tests are conducted, code amended, and tests repeated until the module performs as required

- Testing
  - Test the program before release
  - Tools: white box and black box testing

- Maintenance
  - The program is maintained throughout its life
  - Tools: updates, patches etc.



▲ **Figure 12.1** The program development lifecyle

ANALYSIS STAGE

- Objective: Understand and define the requirements of the software.
- Activities:
  - Gather information from stakeholders.
  - Define functional and non-functional requirements.
  - Analyze user needs and expectations.

METHODS IN ANALYSIS STAGE

- Abstraction : the process of removing or hiding unnecessary detail so that only the important points remain

METHODS IN ANALYSIS STAGE

- Decomposition : breaking down a complex problem into smaller, more manageable sub-problems

Worked example

A learner has been set the task of creating a computer version of the 'noughts and crosses' game where a user plays against the computer.

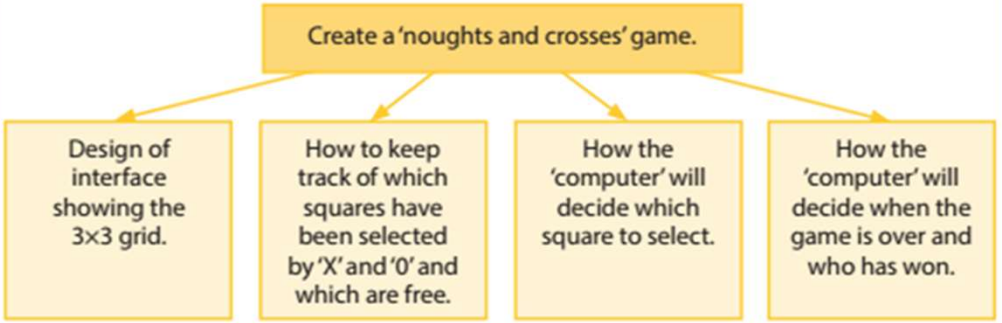
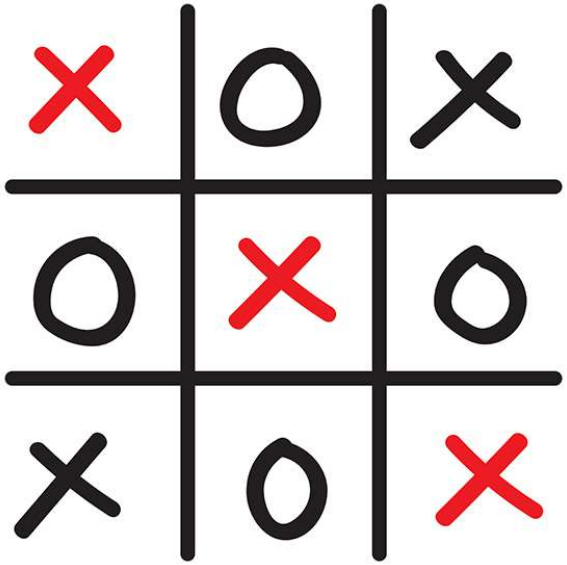


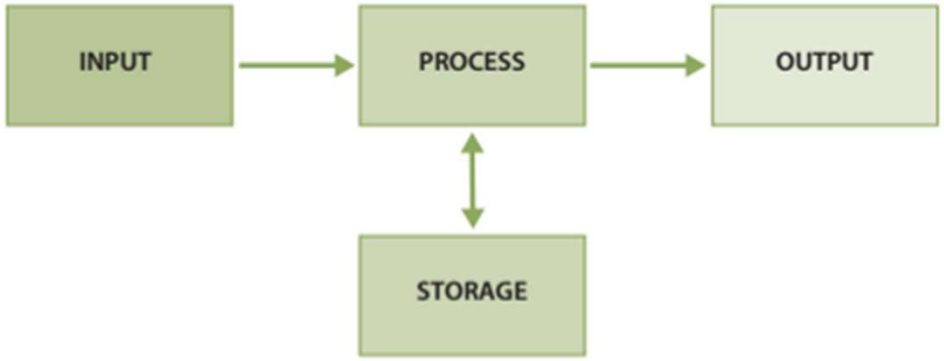
Figure 1.17 Sub-problems to be solved to create a noughts and crosses computer program

The diagram shows some of the sub-problems that must be solved in order to solve the complete problem and create a version of the game.



METHODS IN ANALYSIS STAGE

- Each part, sub-system, sub-routine (they are all the same thing)
  - Consist of:
    - Input: entering data into a computer
    - Process: changing the meaning or format of some data
    - Output: displaying or outputting data that has been processed
    - Storage: data that needs to be stored in files on an appropriate medium for use in the future

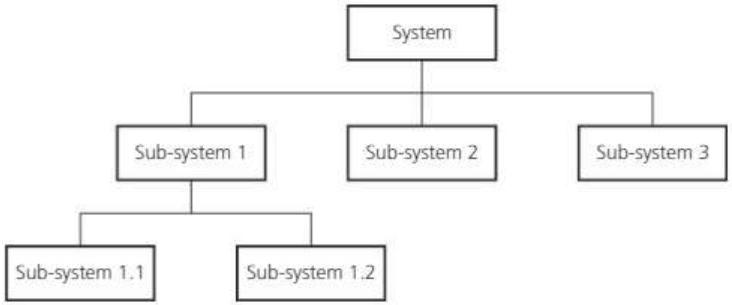


DESIGN STAGE

- Objective: Create a detailed design of the software based on the requirements.
- Activities:
  - Architectural design: Define the overall structure of the software.
  - Detailed design: Specify the data structures, algorithms, and interfaces.
  - User interface design: Design the user interface elements.

METHODS IN DESIGN STAGE

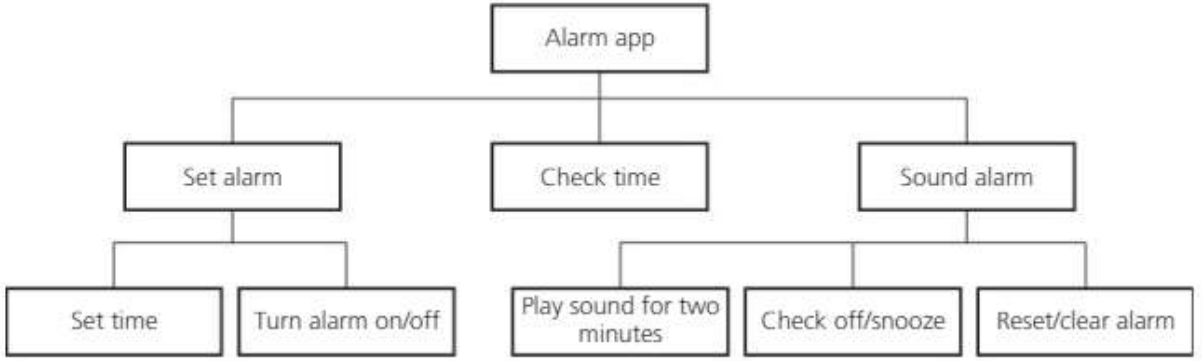
- Structure diagrams
  - It is hierarchical, showing how a computer system solution can be divided into sub-systems with each level giving a more detailed breakdown



▲ **Figure 7.2** Basic structure diagram

Example 2: Alarm app for a smart phone

Consider the alarm app computer system for a smart phone; this could be divided into three sub-systems, setting the alarm, checking for the alarm time, sounding the alarm. These sub-systems could then be further sub-divided; a structure diagram makes the process clearer.



METHODS IN DESIGN STAGE

- Flowcharts : it uses symbols to represent different steps, actions, and decisions
- Pseudocode : it provides a way for developers, designers, and stakeholders to communicate and understand the logic of an algorithm or program without getting bogged down by the syntax of a specific programming language

VALIDATION IN DESIGN STAGE

- Validation is a checking method to see if the input data is sensible before it is accepted into a computer system
- Range Check
  - It checks that the value of a number is between an upper value and lower value
  - For example: checking that percentage marks are between 0 and 100 inclusive

```
OUTPUT "Please enter the student's mark "
REPEAT
INPUT StudentMark
IF StudentMark < 0 OR StudentMark > 100
  THEN
    OUTPUT "The student's mark should be in the range
            0 to 100, please re-enter the mark "
ENDIF
UNTIL StudentMark >= 0 AND StudentMark <= 100
```

VALIDATION IN DESIGN STAGE

- Length check
  - It checks if the input data has an exact number of characters
  - Examples
    - Passwords
    - Mobile numbers
    - Passport numbers

Password has a data type of string and LENGTH is the pseudocode operation that returns a whole number showing the number of characters in the string

```
OUTPUT "Please enter your password of eight
characters "
REPEAT
    INPUT Password
    IF LENGTH(Password) <> 8
      THEN
        OUTPUT "Your password must be exactly eight
                characters, please re-enter "
    ENDIF
UNTIL LENGTH(Password) = 8
```

VALIDATION IN DESIGN STAGE

- Type check
  - It checks the data type of input data if it makes sense
  - Example
    - Number of siblings must be an integer

```
OUTPUT "How many brothers do you have? "
REPEAT
INPUT NumberOfBrothers
IF NumberOfBrothers <> DIV(NumberOfBrothers, 1)
  THEN
    OUTPUT "This must be a whole number, please re-enter"
ENDIF
UNTIL NumberOfBrothers = DIV(NumberOfBrothers, 1)
```

VALIDATION IN DESIGN STAGE

- Presence check
  - It checks to ensure that some data has been entered
  - Example
    - Mobile number must be input when signing up an email account
    - An email must be input when register to a website

```
OUTPUT "Please enter your email address "
REPEAT
INPUT EmailAddress
IF EmailAddress = ""
  THEN
    OUTPUT "*=Required "
ENDIF
UNTIL EmailAddress <> ""
```

VALIDATION IN DESIGN STAGE

- Format check
  - It checks if the input data is the same as pre-defined pattern
  - Example
    - Flight number must begin with two letters and followed by three numbers TG190

```
# Get user input for a flight number
flight_number = input("Enter a flight number: ")

# Check the format of the flight number
if len(flight_number) == 5 and flight_number[:2].isalpha() and flight_number[2:].isdigit():
    print("The flight number is valid.")
else:
    print("Invalid flight number format. Please enter a valid flight number.")
```

VERIFICATION IN DESIGN STAGE

- Verification : It checks that data has been accurately copied from one source to another. This syllabus covers two method; double entry and visual check

- Double entry: enter data twice and compare if they are the same

VERIFICATION IN DESIGN STAGE

- Visual check
  - It is a manual check completed by the user who is entering the data.
  - When the data entry is copied into a computer, the users compares data on the screen with the original source if both versions are the same

CODING STAGE

- We already covered this topic in the coding chapter

TESTING STAGE

- After the program has been developed, the team need to do the testing to ensure that the program is error-free
- This syllabus has 2 methods
  - Test data
  - Trace table (already covered in chapter flowcharts and trace tables)

TESTING STAGE

- Using a test data set : user inputs a data and then compares the actual result with the expected result to see if there is any error.
  - Example, when using a calculator 1+1 the result should be 2 (expected result) but if the actual result is 3, it means there is errors in the calculator
- Test data set; example, a valid score must be between 0-100 inclusive
  - Normal data – This is data that is within the limits of what should be accepted by the program.
    - For example, score = 10, 20, 90
  - Extreme/Boundary data – This is data that is at the upper limit and lower limit of what should be accepted by the program.
    - For example, score = 0 or 100
  - Abnormal data – This is data that are outside the valid range, and it should not be accepted by the program.
    - For example, score = -10 or 110 or ten(invalid data type)

QUESTION

- Explain the difference between a validation check and a verification check.

- Describe, using an example, how data could be verified on data entry

QUESTION

- A validation check is used to make sure that any value that is input is an integer between 30 and 200 inclusive.
- Give one example of each type of test data to check that the validation check is working as intended. Each example of test data must be different.
- Give a reason for each of your choices of test data

QUESTION

- Identify three different ways that the design of a solution to a problem can be presented

QUESTION

- State three different features of a high-level programming language that a programmer could use to make sure that their program will be easier to understand by another programmer.
- Give an example for each feature

QUESTION

(a) **Four** descriptions of validation checks are shown.

Draw **one** line to link each description to the most appropriate check.

Not all checks will be used.

Description	Check
to check that the data entered is an integer	check digit
to check that some data has been entered	format check
to check that the data entered has an appropriate number of characters	length check
to check that an identification number contains no errors	presence check
	type check

[4]

QUESTION

- The input value needs to be stored in the variable Measurement.
- Write pseudocode to perform the double entry check until a successful input is made.