2A.2 Sequence and selection

In Section 1 the basic program structures of sequence, selection and iteration were briefly covered. In this section the use of these structures will be looked at in more detail, with examples of more complex algorithms.

Sequence

All programs have a series of steps to be followed in sequence. Here is an example in which the steps are simple assignment statements:

Example 3

```
OUTPUT "Enter score for Round 1: "
score1 ← USERINPUT
OUTPUT "Enter score for Round 2: "
score2 ← USERINPUT
averageScore ← (score1 + score2) / 2
OUTPUT "The average score is ", averageScore
```

Sometimes the sequence may be a series of calls to different subroutines which perform different tasks. Subroutines (functions and procedures) will be covered in Section 2B.1.

Selection

Before looking at algorithms using the different selection statements available in a programming language, we need to take a closer look at Boolean data types and expressions, since these are used to determine which path through the program will be taken.

Boolean data type

Boolean variables are either **True** or **False**. It makes no sense to perform mathematical operations on them or to compare them to see which is greater. With Boolean variables we use **logical operators** to create **Boolean expressions**. Suppose that A represents some condition, for example x ≤ 10, or speed > 30. Logical operations AND, OR and NOT may be used in Boolean expressions, where:

NOT:	If `A` is `True`, then `NOT A` is `False`
AND:	If `A` is `True` and `B` is `True`, then (`A AND B`) is `True`, otherwise (`A AND B`) is `False`
OR:	If either or both of `A` and `B` is `True`, then (`A OR B`) is `True`, otherwise (`A OR B`) is `False`

Boolean expressions

Boolean expressions are used to control selection statements. For example:

```
IF speed > 30 THEN
   OUTPUT "Control your speed"
ENDIF
```

A complex Boolean expression contains one or more of the operators **AND**, **OR** or **NOT**. For example:

```
IF (X ≤ 10) OR (CurrentCharNum > LengthOfString) THEN …
IF (NOT((A = B) AND (A = C))) THEN OUTPUT "Sides not equal"
```

Q5 Write pseudocode statements to check whether a username entered by the user is equal to either "User1" or "User2". If so, print "Access granted", otherwise print "Access denied".

Nested selection statements

Using a complex Boolean expression is often clearer than using a nested selection structure.

Example 4

Consider an estate agent's program that searches through a file of house details to find ones that match a customer's requirements. In this case the customer wants a house or flat, but it must have more than three bedrooms.

Using a nested IF statement we could write:

```
IF Rooms > 3 THEN
   IF type = "House" THEN
      Output details
   ELSE IF type = "Flat" THEN
      Output details
   ENDIF
ENDIF
```

It is shorter and clearer to write:

```
IF (Rooms > 3) AND ((type = "House") OR (type = "Flat")) THEN
   Output details
ENDIF
```

Notice the extra set of brackets around the second half of the expression. AND takes precedence over OR so without the extra brackets the program would return all the houses with more than three bedrooms as well as any flats, whether they have more than three bedrooms or not.

Writing robust code

Robust code is code which will not result in the program crashing due to an unexpected user input. The pseudocode below would crash for some inputs. Why?

```
num1 ← USERINPUT
num2 ← USERINPUT
OUTPUT num1/num2
```

The algorithm needs to be amended so that it will not crash whatever the user enters.

```
num1 ← USERINPUT
num2 ← USERINPUT
IF (num2 = 0) THEN
   OUTPUT "Cannot divide by 0"
ELSE
   OUTPUT num1 / num2
ENDIF
```

Example 5

A room is to be carpeted using carpet that is 4m wide. The program asks the user to enter the room dimensions, and if the width is greater than 4, outputs `"Carpet not wide enough"`. Otherwise, it calculates the length of carpet required by adding 5% to the length of the room.

The following algorithm has been written:

```
roomLength ← USERINPUT
roomWidth ← USERINPUT
IF roomWidth > 4 THEN
   OUTPUT "Carpet not wide enough"
ELSE
   carpetLength ← roomLength * 1.05
   OUTPUT "Length of carpet required = ", carpetLength
ENDIF
```

The algorithm could still give the wrong answer if the user entered a width greater than 4, and a length less than the width, as it assumes that the user always enters the shortest dimension as the width. The program needs to check for this. Here is the rewritten algorithm:

```
roomLength ← USERINPUT
roomWidth ← USERINPUT
IF (roomWidth > 4) AND (roomLength < roomWidth) THEN
   temp ← roomLength
   roomLength ← roomWidth
   roomWidth ← temp
ENDIF
IF roomWidth > 4 THEN
   OUTPUT "Carpet not wide enough"
ELSE
   carpetLength ← roomLength * 1.05
   OUTPUT "Length of carpet required = ", carpetLength
ENDIF
```

Q6 How much carpet does the algorithm calculate is required if the user enters:

(a) a length of 2 and a width of 4?

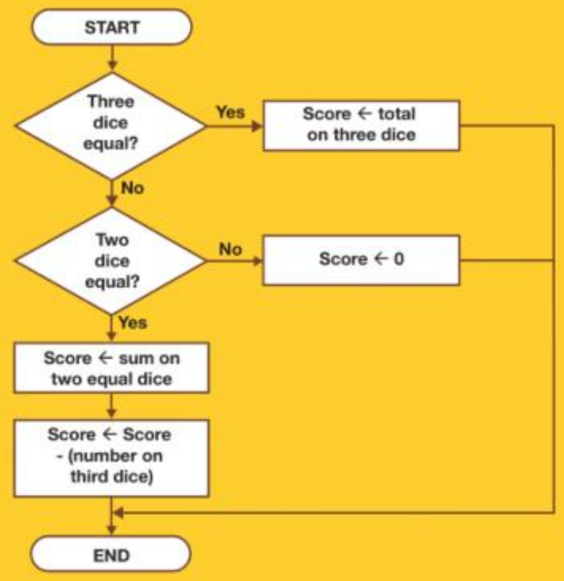
(b) a length of 3 and a width of 5?

Q7 In Section 1, you learned how to write complex selection statements using a nested `IF` statement

Write an algorithm to solve the following problem.

- If a student gets A* in GCSE Maths, he or she will be advised to consider taking Further Maths at A Level
- If a student gets A in GCSE Maths, he or she will be advised to consider taking Maths at A Level
- If a student gets B in GCSE Maths, he or she will be advised to consider taking Maths at AS Level
- Otherwise, they will be advised not to continue with Maths.

Q8 Julie has written an algorithm in the form of a flowchart for a dice game played with three dice. When the player rolls the dice, they are given points according to the rules shown in the flowchart.


START
Three dice equal?
Yes
Score ← total on three dice
No
Two dice equal?
No
Score ← 0
Yes
Score ← sum on two equal dice
Score ← Score - (number on third dice)
END

(a) Show the value of the score if the dice rolled are: (i) 2 4 6 (ii) 5 5 5 (iii) 2 2 3

(b) State a set of numbers which will result in a negative score.

(c) Write a selection statement which will test if all three dice are equal.