11 TESTING AND EVALUATION

DID YOU KNOW?

The Ariane 5 space rocket was launched by the European Space Agency in 1996. Its flight lasted 39 seconds before it exploded. Data about velocity was stored in 16-bit variable. This had been fine for the Ariane 4 but the 5 model was much faster and the value was too large. This one software error set of a chain of events leading to self-destruction.

However carefully an algorithm has been created and coded into a program, there will always be mistakes and errors. Before a program goes live, it should be thoroughly tested and evaluated to ensure it meets the requirements.

LEARNING OBJECTIVES

- Describe different types of error in programs
- Design and use test plans and test data
- Determine what value a variable will hold at a given point in a program
- Evaluate the strengths and weaknesses of a program and suggest improvements

LOGIC ERRORS

EXTEND YOUR KNOWLEDGE

Research other problems and disasters that have been caused by software failures.

DID YOU KNOW?

In computer programming the order of precedence (the order in which you do each calculation) is the same as in mathematics and science – **BIDMAS**.

This is how $3^2 \times 9 + (5 - 2)$ would be evaluated.

Brackets	$32 \times 9 + (3)$
Indices	$9 \times 9 + (3)$
Division	
Multiplication	$81 + (3)$
Addition	84
Subtraction	

To calculate 24/3 – 2, the division would be calculated before the subtraction.

24/3 = 8
8 – 2 = 6

Logic errors occur when the thinking behind an algorithm is incorrect so that the output isn't what is expected or intended. Ideally, logic errors should be identified and fixed at the design stage.

The following algorithm is intended to work out whether a learner has passed a test. Learners need a score of 80 or above to pass. However, a logic error in the algorithm means that it produces an incorrect and unexpected result.

```
IF testScore <= 80 THEN
    SEND 'Pass' TO DISPLAY
ELSE
    SEND 'Failed' TO DISPLAY
END IF
```

WORKED EXAMPLE

Here is an algorithm to find the average of two numbers.

```
RECEIVE number1 FROM KEYBOARD
RECEIVE number2 FROM KEYBOARD
SET average TO number1 + number2 / 2
SEND average TO DISPLAY
```

This seems logical. Two numbers are input, they are added together and then they are divided by 2.

However, if this algorithm was given 12 and 6 as the two numbers it would return 15 as the average instead of 9. There is a logic error.

Instead of adding the two numbers and then dividing by 2, as the developer intended, it is dividing the second number by 2 and then adding the result to the first number.

The developer should have written the third line as:

```
SET average TO (number1 + number2) / 2
```

SKILLS PROBLEM SOLVING

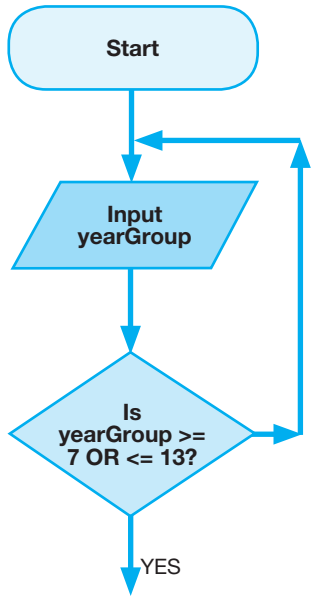
ACTIVITY 28

LOGIC ERRORS

This is part of a larger algorithm designed to ensure that input data falls within a certain range. It is part of a school management system. It checks that the 'year group' entry is acceptable. It has students aged 11 to 18 years with year groups of 7 to 13.

The staff using the system congratulated themselves on never making an error when entering the year group. However, when student lists were printed out, they immediately received complaints.

With a partner, discuss this algorithm and find the logic error in the algorithm?



▲ **Figure 2.4** Flowchart showing an error in an algorithm

WORKED EXAMPLE

Study this algorithm.

```
WHILE index < 10 DO
      SET index TO 1
      SEND index TO DISPLAY
      SET index TO index + 1
END WHILE
```

The expected output is 1, 2, 3, 4, 5, 6, 7, 8, 9.

But there is a logic error. The variable 'index' is initialised in the wrong place. It should be done before the start of the `WHILE` loop.

This algorithm would loop forever because at each turn in the loop the variable index is set to 1. It will never reach 10. This is an

SUBJECT VOCABULARY

infinite loop a loop that is never-ending since the condition required to **terminate** the loop is never reached

GENERAL VOCABULARY

terminate to cause something to end or stop

SKILLS REASONING, PROBLEM SOLVING

example of an '**infinite loop**'. The algorithm should have been written as shown below.

```
SET index TO 1
WHILE index < 10
      SEND index TO DISPLAY
      SET index TO index + 1
END WHILE
```

ACTIVITY 29

FINDING AND CORRECTING ERRORS

Find and correct the errors in these algorithms.

Example 1

```
SET index TO 1
WHILE index < 10
    SEND index TO DISPLAY
END WHILE
```

Example 2

```
SET index TO 1
WHILE index < 10
    SEND index TO DISPLAY
    SET index TO index - 1
END WHILE
```

Example 3

```
SET index TO 1
WHILE index < 1
    SEND index TO DISPLAY
    SET index TO index + 1
END WHILE
```

TRACE TABLES

The formal way of checking the logic of an algorithm is to use a **trace table**.

SUBJECT VOCABULARY

logic error an error in an algorithm that results in incorrect or unexpected behaviour

trace table a technique used to identify any logic errors in algorithms. Each column represents a variable or output and each row a value of that variable

WORKED EXAMPLE

Create a trace table for the following algorithm written in Pearson Edexcel pseudocode.

```
SET number TO 3
FOR index FROM 1 TO 5 DO
    SET number1 TO number * index
    SET number2 TO number1 * 2
    IF number2 > 20 THEN
        SEND number2 TO DISPLAY
    END IF
END FOR
```

This algorithm can be traced as in Table 2.12.

NUMBER	INDEX	NUMBER1	NUMBER2	OUTPUT
3				
3	1	3	6	
3	2	6	12	
3	3	9	18	
3	4	12	24	24
3	5	15	30	30

▲ **Table 2.12** Trace table

The value of the variable number remains at 3 throughout, but as the index increases from 1 to 5, then so do the values of `numberl` and `number2`.

When the value of `number2` is greater than 20, its value is output.

WORKED EXAMPLE

Here is a program.

Python

```
y = 2
for x in range(1, 7):
      y = y + x
print(y)
```

Java

```
class Main {
  public static void main(String[] args) {
    int y = 2;
    for(int x = 1; x < 7; x++) {
      y += x;
    }
    System.out.println(y);
  }
}
```

C#

```
int number = 3;
int number1, number2;

for (int index = 1; index <= 5; index++)
{
    number1 = number * index;
    number2 = number1 * 2;
    if (number2 > 20)
    {
        Console.WriteLine(number2);
    }
}
```

X	Y	OUTPUT	EXPLANATION
1	2		When the loop starts, X becomes 1 and Y already is equal to 2.
2	3		When X is incremented to 2, Y is equal to 3 (2+1) from the previous loop.
3	5		When X is incremented to 3, Y= 3 + 2 from the previous loop.
4	8		Explanations as above.
5	12		
6	17		
6	23	23	The final value of Y is output.

▲ **Table 2.13** Trace table of the programs above

ACTIVITY 30

Complete a trace table for this program written in a high-level language.

Python

```
number1 = 2
number2 = 3
for index in range(1, 6)
      number1 = number1 * index
      number2 = number2 + number1
```

Java

```
class Main {
  public static void main(String[] args) {
    int number1 = 2;
    int number2 = 3;
    for(int index = 1; index < 6; index++) {
      number1 = number1 * index;
      number2 = number2 + number1;
    }
  }
}
```

C#

```
int number1 = 2;
int number2 = 3;
for (int index = 1; index < 6; index++)
{
      number1 = number1 * index;
      number2 = number2 + number1;
}
```

SKILLS PROBLEM SOLVING

ACTIVITY 31

The following program is intended to count the number of female learners in a class.

Python

```
gender = ['M', 'M', 'F', 'M', 'F', 'F', 'M', 'F', 'M', 'F']
length = len(gender)
count = 0
for index in range(length):
      if gender[index] 'F':
            count = count + 1
```

Java

```
class Main {
  public static void main(String[] args) {
    String[] gender = {"M", "M", "F", "M", "F", "F",
"M", "F", "M", "F"};
    int length = gender.length;
    int count = 0;
    for(int index = 0; index < length; index++) {
      if(gender[index].equals("F")) {
        count++;
      }
    }
  }
}
```

C#

```
int length, count;
char[] gender;
gender = new char[]{ 'M', 'M', 'F', 'M', 'F', 'F', 'M',
'F', 'M', 'F' };

count = 0;
length = gender.Length;
for (int index = 0; index < length; index++)
{
  if (gender[index] == 'F')
  {
    count += 1;
  }
}
```

1 Create and complete a trace table for this algorithm.
2 Create a trace table for the algorithm below using the following sample data: 3, 12, 21, 28, 0.

Python

```
total = 0
number = int(input('Please enter the number')
while number > 0:
      total = total + number
      number = int(input('Please enter the number')
print total
```

Java

```
import java.util.*;
class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    int total = 0;
    System.out.print("Please enter the number: ");
    int number = scanner.nextInt();
    while(number > 0) {
      total += number;
      number = scanner.nextInt();
    }
    System.out.println(total);
  }
}
```

C#

```
int total = 0;
int number;
string numberString;

Console.WriteLine("Please enter the number");
numberString = Console.ReadLine();
number = int.Parse(numberString);

while (number > 0)
{
  total += number;
  Console.WriteLine("Please enter the number");
  numberString = Console.ReadLine();
  number = int.Parse(numberString);
}
Console.WriteLine(total);
```

SYNTAX ERRORS

Syntax errors occur when the grammar rules of a programming language are not followed.

They prevent the code from being compiled or translated.

Examples of syntax errors are:

- writing 'prnit' instead of 'print'
- missing out a closing bracket
- missing out quotation marks.

RUNTIME ERRORS

SUBJECT VOCABULARY

runtime error an error that occurs while the program is running – the operation the computer is asked to do is impossible to execute

Runtime errors occur during program execution and are the most difficult to predict and spot.

This program is designed to take two numbers, divide the first number by the second number and output the result. It will work as intended at least some of the time. However, if the user entered 5 and 0, a runtime error would occur because it is impossible for the computer to divide 5 by 0.

Python

```
firstNumber = int(input('Please enter the first number')
secondNumber = int(input('Please enter the second
number')
result = firstNumber / secondNumber
print(result)
```

Java

```
import java.util.*;
class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Please enter the first number: ");
    int firstNumber = scanner.nextInt();
    System.out.print("Please enter the second number: ");
    int secondNumber = scanner.nextInt();
    scanner.close();

    // do the division (converting all integers to reals)
    double result = (double)firstNumber / (double)
secondNumber;
    System.out.println(result);
  }
}
```

C#

```
int firstNumber, secondNumber, result;
string firstNumberString, secondNumberString;

Console.WriteLine("Please enter the first number");
firstNumberString = Console.ReadLine();
firstNumber = int.Parse(firstNumberString);

Console.WriteLine("Please enter the second number");
secondNumberString = Console.ReadLine();
secondNumber = int.Parse(secondNumberString);

result = firstNumber / secondNumber;
Console.WriteLine(result);
```

ERROR SUMMARY

This table gives a summary of the three types of error you are likely to encounter.

TYPE OF ERROR	DESCRIPTION
Logic	The program seems to run normally; however, there is an error in the logic of the program, which means it does not produce the result you expect.
Syntax	Syntax refers to the rules of the programming language. A syntax error means that part of the code breaks the rules of the language, which stops it running.
Runtime	An error that occurs when the computer tries to run code that it cannot execute.

▲ **Table 2.14** Summary of the three types of error you are likely to encounter

USING AN INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

SUBJECT VOCABULARY

Integrated Development Environment (IDE) a package that helps programmers to develop program code. It has a number of useful tools, including a source code editor and a debugger

debugger a computer program that assists in the detection and correction of errors in other computer programs

You probably already have first-hand experience of using an **Integrated Development Environment (IDE)** when writing code. It's definitely worth taking some time to get to know the IDE that comes with the language you are using. Useful features such as syntax highlighting, code auto complete and auto indent will help to make your programming experience far less stressful, especially at the beginning.

One of the most useful features of an IDE is the **debugger**. One of its tasks is to flag up syntax errors in the code and issue helpful error messages. It is really important that you get lots of practice interpreting error messages and fixing errors in your code.

HINT

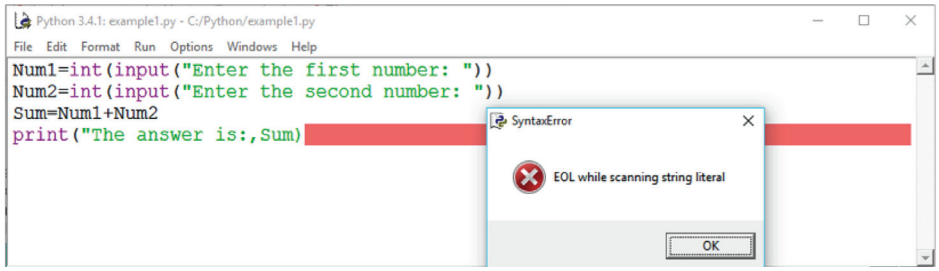
Here are some error messages that you are likely to see in popular IDEs.

IDLE (a popular Python IDE)

- `SyntaxError:` invalid syntax – *part of the code breaks one of the rules of the programming language.*
- `IndentationError:` expected an indented block – *statements after a colon must be indented.*
- `TypeError:` Can't convert 'int' object to str implicitly – *trying to join a string and an integer together.*
- `ZeroDivisionError:` integer division or modulo by zero – *trying to divide a value by 0.*
- `NameError:` name is not defined – *referring to a variable or subprogram that does not exist.*

Eclipse (a popular Java IDE)

- ; expected – *each statement should end with a semicolon.*
- Cannot find symbol – *referring to a variable or subprogram that does not exist.*
- Incompatible types – *trying to mix data of different types, for example a string and an integer.*



▶ **Figure 2.5** A syntax error flagged up by an IDE in Python code

THE TEST PLAN

At the start of a programming project, it is crucial to make a list of the requirements that the program is expected to meet. Throughout the development phase of the project, you should regularly refer back to this list of requirements to check that you are on track to achieve them.

Deciding how to test the finished program to make sure that it fully meets the requirements can't be left to the last moment either. As part of the design stage, you should create a test plan listing the tests you will carry out, the data that will be used for each test and the expected result.

MARKS TO GRADES

```
RECEIVE examMark FROM (INTEGER) KEYBOARD
IF examMark >= 80 THEN
   SEND 'A' TO DISPLAY
ELSE
   IF examMark >= 70 THEN
      SEND 'B' TO DISPLAY
   ELSE
      IF examMark >= 60 THEN
         SEND 'C' TO DISPLAY
      ELSE
         IF examMark > 0 THEN
            SEND 'D' TO DISPLAY
         ELSE
            SEND 'FAIL' TO DISPLAY
         END IF
      END IF
   END IF
END IF
```

This algorithm converts an exam mark into a grade. It should only accept marks between 0 and 100.

The test plan extract in Table 2.15 shows some of the tests that have been planned for the finished program to ensure that it meets all the requirements.

TEST NO.	PURPOSE OF THE TEST	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	ACTION NEEDED/ COMMENTS
1	To check correct conversion of valid mark	0 55 65 75 85	'FAIL' 'D' 'C' 'B' 'A'		
2	To check correct conversion of boundary mark	0 1, 59 60, 69 70, 79 80, 100	'FAIL' 'D' 'C' 'B' 'A'		
3	To check response to erroneous mark	–5 105	Error message Error message		

▲ **Table 2.15** Test plan extract

Only the first four columns of the test plan table can be completed at the design stage. The remaining columns are filled in when the program is complete.

At the start, it's unlikely that you'll know every test that will be needed to ensure the program works as intended. The test plan is not a fixed document. If additional tests are required, they should be added to the test plan.

If a test produces the expected result, you can simply write 'None' in the 'Action needed/comments' column. If, however, that is not the case then you should instead note what went wrong and what you did to put it right.

It is important to select suitable test data for the tests. Test data falls into three different categories: normal, boundary and erroneous.

Normal data	Data that is well within the limits of what should be accepted by the program.	Test 1 uses normal data to check if marks are converted into grades correctly (e.g. a mark of 65 should be converted to a grade C).
Boundary data	Data that is at the outer limits of what should be accepted by the program.	Test 2 uses boundary data to check that the program works correctly with marks at the upper and lower boundaries (e.g. a mark of 60 and a mark of 69 should both convert to a grade C).
Erroneous data	Data that should not be accepted by the program.	Test 3 uses erroneous data to check that the program does not accept it.

▲ **Table 2.16** Test data categories

Testing is just as important as writing code because it ensures that the finished program works correctly and fully meets the requirements. You should use a 'bottom up' approach to testing (i.e. test each subprogram as you develop it and then test the whole program once it is finished).

Here is the completed test plan for the grade calculator program. As you can see, the expected result was not produced when the program was tested with erroneous data. A range check had to be added to the program to ensure that only marks between 0 and 100 can be entered.

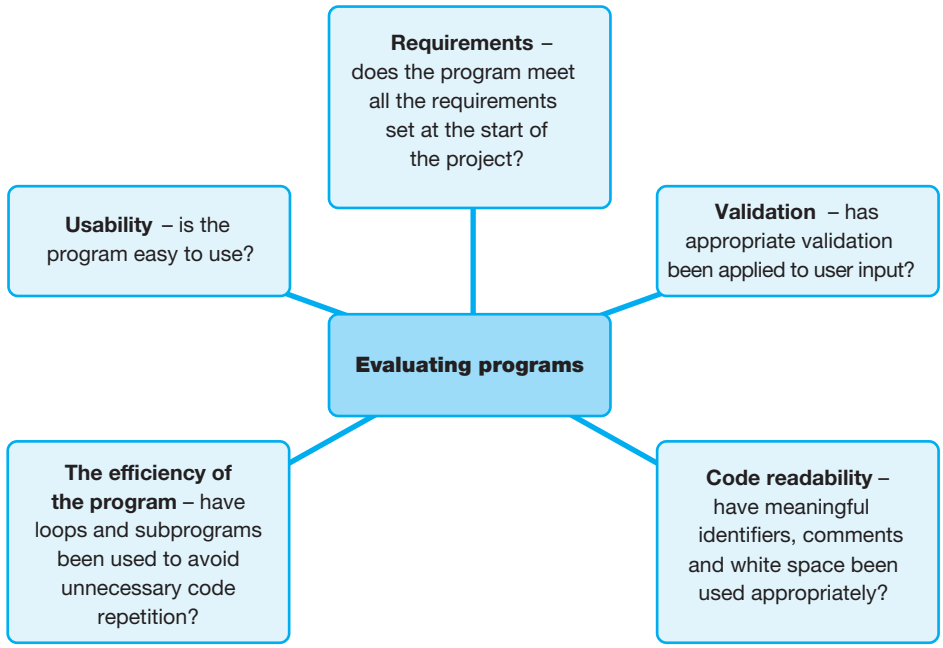
TEST NO.	PURPOSE OF THE TEST	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	ACTION NEEDED/ COMMENTS
1	To check correct conversion of valid mark	0 55 65 75 85	'FAIL' 'D' 'C' 'B' 'A'	'FAIL' 'D' 'C' 'B' 'A'	None
2	To check correct conversion of boundary mark	0 1, 59 60, 69 70, 79 80, 100	'FAIL' 'D' 'C' 'B' 'A'	'FAIL' 'D' 'C' 'B' 'A'	None
3	To check response to erroneous mark	–5 105	Error message Error message	'FAIL' 'A'	A range check has been added to ensure that only valid marks can be added.

▲ **Table 2.17** The completed test plan for the grade calculator program

After you have carried out all of the tests and made all the necessary changes, the program should be retested. This is to ensure that the improvements you have made haven't introduced any new errors into the program.

EVALUATING PROGRAMS

You need to be able to identify the strengths and weaknesses of your own programs as well as those created by other programmers. This will enable you to identify techniques that work well and aspects that could be improved.



▶ **Figure 2.6** Consider the aspects shown when evaluating a program

SKILLS PROBLEM SOLVING

ACTIVITY 32

CREATING A GUESSING GAME ALGORITHM

1. Develop an algorithm for a simple guessing game. The algorithm must:
   - **a** generate a random number between 1 and 6
   - **b** ask the user to input a number between 1 and 6
   - **c** reject any values outside the acceptable range
   - **d** display 'Well Done' if the user guesses correctly or 'Try Again' if their guess is incorrect.
2. Develop a test plan for this guessing game program.
3. Can you write the guessing game program in the high-level programming language you are studying?
4. Test your program by carrying out the tests you planned. Ensure you update your test plan after completing each test.

CHECKPOINT

Strengthen

SKILLS CRITICAL THINKING, REASONING

S1 What are the three types of error associated with program development? Can you identify the stage(s) of development at which they are most likely to occur?

SKILLS CRITICAL THINKING

S2 Outline the function of a trace table.

SKILLS REASONING

S3 What are the features of an IDE that help programmers write error-free code?

SKILLS CRITICAL THINKING

S4 What is the function of a test plan in program development?

SKILLS CRITICAL THINKING

S5 What is meant by normal, boundary and erroneous data?

SKILLS REASONING

S6 Why might it be necessary to retest a program once all the planned tests are completed?

Challenge

SKILLS PROBLEM SOLVING

C1 Think of a problem which could be solved using a computer program. What are the requirements for the program? Create a solution and draw up a test plan for it.

SKILLS PROBLEM SOLVING

C2 Develop and implement the program. Conduct your planned tests and make any necessary changes to your program, ensuring your test plan is kept up to date.

SKILLS REASONING

C3 Reflect on your program and provide an evaluation.

How confident do you feel about your answers to these questions? If you're not sure you answered them well, try completing Activities 28–32.

SUMMARY

- Logic errors occur when there is an error in the logic of the code, causing the program to produce an unexpected result.
- A syntax error occurs when part of the code breaks the rules of the programming language.
- A runtime error occurs while the program is running and it is asked to do something that is impossible to do.
- Trace tables can be used to manually trace the execution of an algorithm, allowing you to track the changes in variable values.
- Before creating a program, it is important to produce a test plan that outlines how the final program will be tested to ensure it meets the requirements.
- Evaluating a program involves considering its strengths and weaknesses and identifying areas for improvement.

UNIT QUESTIONS

SKILLS CRITICAL THINKING, REASONING AO2

1 a Identify the line number(s) that show one example of each of these structural components in the program below: **(6)**

- variable initialisation
- type declaration
- selection
- iteration
- data structure
- subprogram.

```
1  SET valuesArray TO [3, 9, 12, 16, 4, 98]
2  REAL max
3
4  FUNCTION maxCalc(values)
5  BEGIN FUNCTION
6      SET length TO LENGTH(values)
7      SET max TO 0
8      FOR index = 0 TO length - 1 DO
9          IF values[index] > max THEN
10         SET max TO values[index]
11         END IF
12     END FOR
13 RETURN max
14 END FUNCTION
15
16 SET max TO maxCalc(valuesArray)
   SEND max TO DISPLAY
```

b Identify the line number where a subprogram call is made. **(1)**

HINT

Question 1 is testing your ability to identify the seven key structural components of a program. Make sure you have identified the line numbers in which the components can be found.

SKILLS CRITICAL THINKING, REASONING AO3

2 Describe what this algorithm does. **(2)**

```
SET scores TO [45, 67, 34, 98, 52]
SET length TO LENGTH(scores)
SET count TO 0
FOR index FROM 0 TO length – 1 DO
    IF scores[index] >= 50 THEN
        SET count TO count + 1
    END IF
END FOR
```

SKILLS CRITICAL THINKING, PROBLEM SOLVING, REASONING — A02

3 Draw and complete a trace table for this algorithm with these column headings: **(5)**

- length
- count
- index
- scores[index].

HINT

Question 3 tests your ability to trace an algorithm using a trace table. Remember each variable change should be recorded and each time a line of code alters the value of a variable or variables a new row of the trace table should be completed.

SKILLS CRITICAL THINKING, REASONING — A02

4 Open file Q01a. Answer these questions about the code.

a State the name of a user-defined subprogram. **(1)**
b State the name of one in-built subprogram. **(1)**
c State the names of one input parameter. **(1)**
d State the name of a global variable. **(1)**
e State the name of a local variable. **(1)**
f State the line number of the command that 'calls' the variable. **(1)**

HINT

These questions are asking you to 'state' various elements in the program. You do not have to describe them or explain how they function.

SKILLS CRITICAL THINKING, PROBLEM SOLVING, REASONING — A02, A03

5 Open file Q02a. Answer these questions about the code.
A data structure has been used to store the results of a survey to find favourite brands of automobiles.

They have been stored in ascending order.

a Name this type of data structure. **(1)**
b The user is asked to enter the name of a brand and their input is converted into upper case. Explain why this is done. **(2)**
c Complete the program to search for the brand name entered in the data structure. If it is present, then the program should inform the user of its position, e.g. Audi is in position 1.
The user should also be informed if it is not in the list.
Save your completed program as Q02b. **(6)**

HINT

This question is asking you to traverse a list to find a particular item. When you print out the result text and variables have to be concatenated.

SKILLS PROBLEM SOLVING — A03

6 Open file Q03a.

It shows a list of users and their (not very strong) passwords stored in a list. Complete the program so that a user can enter their user name. If the name exists, then they should be asked for their password. They should be informed of the following:

- if the username and password are correct
- if the name they entered is not recognised. **(10)**

HINT

The question requires you to use a loop and selection.