AQA

AS
COMPUTER SCIENCE

Paper 1

Monday 5 June 2017 Morning Time allowed: 1 hour 45 minutes

Materials
For this paper you must have access to:
- a computer
- a printer
- appropriate software
- the Electronic Answer Document
- an electronic version and a hard copy of the Skeleton Program
- an electronic version of the Data File
- an electronic version and a hard copy of the Preliminary Material.

You must **not** use a calculator.

Instructions
- Type the information required on the front of your Electronic Answer Document.
- Before the start of the examination make sure your **Centre Number**, **Candidate Name** and **Candidate Number** are shown clearly **in the footer** of every page (not the front cover) of your Electronic Answer Document.
- Enter your answers into the Electronic Answer Document.
- Answer **all** questions.
- Save your work at regular intervals.

Information
- The marks for questions are shown in brackets.
- The maximum mark for this paper is 75.
- No extra time is allowed for printing and collating.
- The question paper is divided into **three** sections.

Advice
You are advised to allocate time to each section as follows:
Section A – 20 minutes; **Section B** – 20 minutes; **Section C** – 65 minutes.

At the end of the examination
Tie together all your printed Electronic Answer Document pages and hand them to the Invigilator.

Warning
It may not be possible to issue a result for this paper if your details are not on every page of your Electronic Answer Document.

Section A

You are advised to spend no more than **20 minutes** on this section.

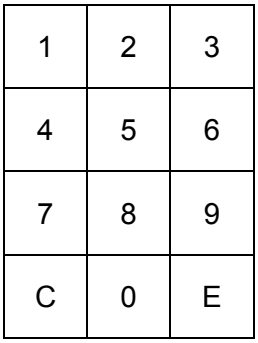
Enter your answers to **Section A** in your Electronic Answer Document. You **must save** this document at regular intervals.

Question 03 in this section asks you to write program code **starting from a new program/project/file**.

You are advised to save your program at regular intervals.

0 1 A hotel provides a safety deposit box in guest rooms. The safety deposit box has a keypad with twelve buttons, as shown in **Figure 1**.

Figure 1



The safety deposit box operates as follows:

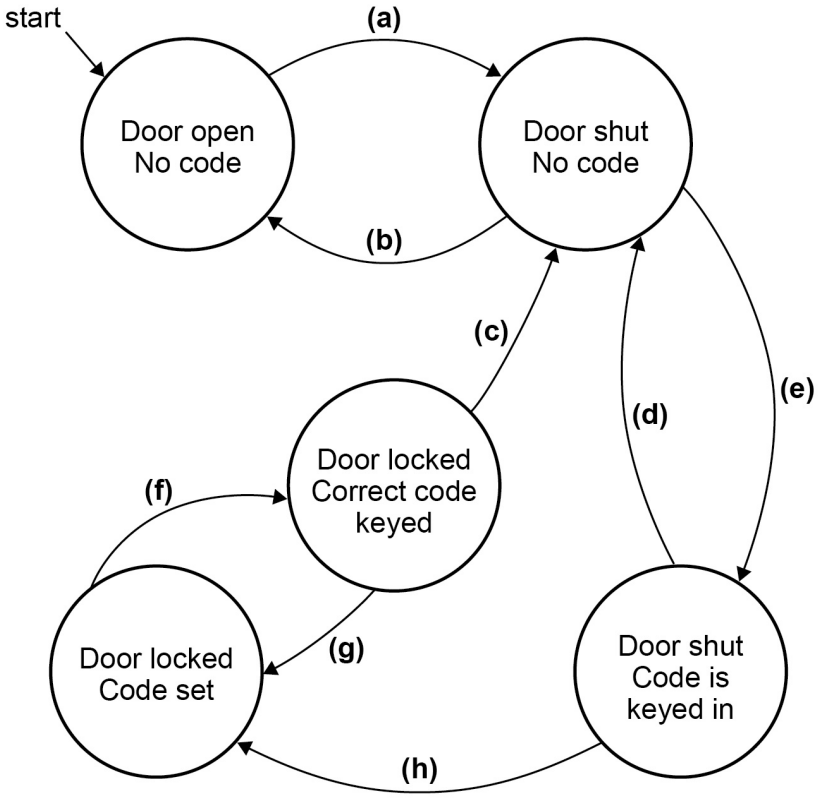
The buttons with digits 0 to 9 enable the guest to set their own code. Button C cancels any digits entered. Button E is the Enter key.

- To close the safety deposit box:
  - push the door shut
  - key in a new 4-digit code (guest's choice)
  - press the Enter key (this sets the code and locks the door).
- To open the safety deposit box:
  - key in the correct 4-digit code (previously chosen by the guest)
  - press the Enter key (this also deletes the stored code)
  - pull open the door.
- Pressing the keypad has no effect, except when keying in a code.

Figure 2 shows a partially complete state transition diagram that represents the operation of the safety deposit box. The events are labelled **(a)** to **(h)**.

Note the state transition diagram does not show what happens if an incorrect code is keyed in.

Figure 2



0 1 . 1 In **Table 1** indicate which label(s), **(a)** to **(h),** represent(s) which event. Two labels have to be assigned to some of the events.

Complete **Table 1** by filling in the unshaded cells with the correct labels from **Figure 2**. A label **must** only be used once.

Table 1

Event	Label(s): (a) to (h)
Correct code keyed	
Door pulled open	
Door pushed shut	
New code keyed	
Press C	
Press E	

Copy the contents of all the unshaded cells in **Table 1** into your Electronic Answer Document.

[4 marks]

Turn over ►

0 2 The algorithm represented using pseudo-code in **Figure 3** describes a method to convert two hexadecimal numbers into decimal. The subroutine `ToDecimal` used in **Figure 3** is shown in **Figure 4** and the built-in subroutine `ASCII` is explained in **Table 2**.

Figure 3

```
FOR Count ← 1 TO 2
  INPUT HexString
  Number ← 0
  FOR EACH HexDigit IN HexString
    Value ← ToDecimal(HexDigit)
    Number ← Number * 16 + Value
  ENDFOR
  OUTPUT Number
ENDFOR
```

The `FOR EACH` command steps through each character in a string working from left to right.

Figure 4

```
SUBROUTINE ToDecimal(HexDigit)
  IF HexDigit = "A" THEN
    Value ← 10
  ELSEIF HexDigit = "B" THEN
    Value ← 11
  ELSEIF HexDigit = "C" THEN
    Value ← 12
  ELSEIF HexDigit = "D" THEN
    Value ← 13
  ELSEIF HexDigit = "E" THEN
    Value ← 14
  ELSEIF HexDigit = "F" THEN
    Value ← 15
  ELSEIF HexDigit IN ["0", "1", ..., "9"] THEN
    Value ← ASCII(HexDigit) - 48
  ELSE
    Value ← -1
  ENDIF
  RETURN Value
ENDSUBROUTINE
```

Table 2

Subroutine used in Figure 4	Description
`ASCII(Char)`	Returns the ASCII code of the character passed as a parameter. Example: `ASCII("1")` returns `49`

0 2 . 1 Complete **Table 3** by hand-tracing the algorithm in **Figure 3**. Use `"A2"` and `"1G"` as input strings. You may not need to use all the rows in **Table 3**.

Table 3

Count	HexString	Number	HexDigit	Value	Output

Copy the contents of all the unshaded cells in **Table 3** into your Electronic Answer Document.

[5 marks]

0 2 . 2 Explain how the algorithm in **Figure 3** has attempted to deal with the conversion of `"1G"` into decimal **and** why this method is not fully effective.

[2 marks]

Turn over for the next question

0 3 The algorithm represented using pseudo-code in **Figure 5** describes a method to find the greatest common factor (GCF) of two whole numbers (integers) entered by the user.

Figure 5

```
OUTPUT "Enter a whole number: "
INPUT Number1
OUTPUT "Enter another whole number: "
INPUT Number2
Temp1 ← Number1
Temp2 ← Number2
WHILE Temp1 ≠ Temp2
  IF Temp1 > Temp2 THEN
    Temp1 ← Temp1 - Temp2
  ELSE
    Temp2 ← Temp2 - Temp1
  ENDIF
ENDWHILE
Result ← Temp1
OUTPUT Result, " is GCF of ", Number1, " and ", Number2
```

What you need to do:

Task 1
Write a program to implement the algorithm in **Figure 5**.

Task 2
Test the program by showing the result of entering `12` and then `39`.

Evidence that you need to provide
Include the following in your Electronic Answer Document.

0 3 . 1 Your PROGRAM SOURCE CODE for **Task 1**.

[6 marks]

0 3 . 2 SCREEN CAPTURE(S) showing the test described in **Task 2**.

[1 mark]

The algorithm copies the values of `Number1` and `Number2` into `Temp1` and `Temp2` respectively.

0 3 . 3 Explain why the `WHILE` loop was written using `Temp1` and `Temp2` instead of `Number1` and `Number2`.

[1 mark]

Section B

You are advised to spend no more than **20 minutes** on this section.

Enter your answers to **Section B** in your Electronic Answer Document. You **must save** this document at regular intervals.

These questions refer to the **Preliminary Material** and the **Skeleton Program**, but do **not** require any additional programming.

Refer **either** to the **Preliminary Material** issued with this question paper **or** your electronic copy.

0 4 State the name of an identifier for:

0 4 . 1 a variable that is used to store a Boolean value.

[1 mark]

0 4 . 2 a user-defined subroutine that returns an integer value.

[1 mark]

0 4 . 3 an array or list variable.

[1 mark]

0 4 . 4 a local variable used to store a string value.

[1 mark]

Turn over for the next question

0 5

0 5 . 1 Explain the benefits of defining `FIELDLENGTH` and `FIELDWIDTH` as named constants instead of using the actual values in the code.

[2 marks]

0 5 . 2 Explain the purpose of the first selection structure in the subroutine `SeedLands`.

[1 mark]

0 5 . 3 The simulation is to be refined to include another level of drought: a minor drought.

There is a 1 in 3 chance of a severe drought, and a 1 in 3 chance of a minor drought. If there is a minor drought, then one in four plants will die.

Describe the changes that need to be made to the subroutine `SimulateSummer` in order to simulate a minor drought.

You are **not** expected to make any changes to the **Skeleton Program**.

[3 marks]

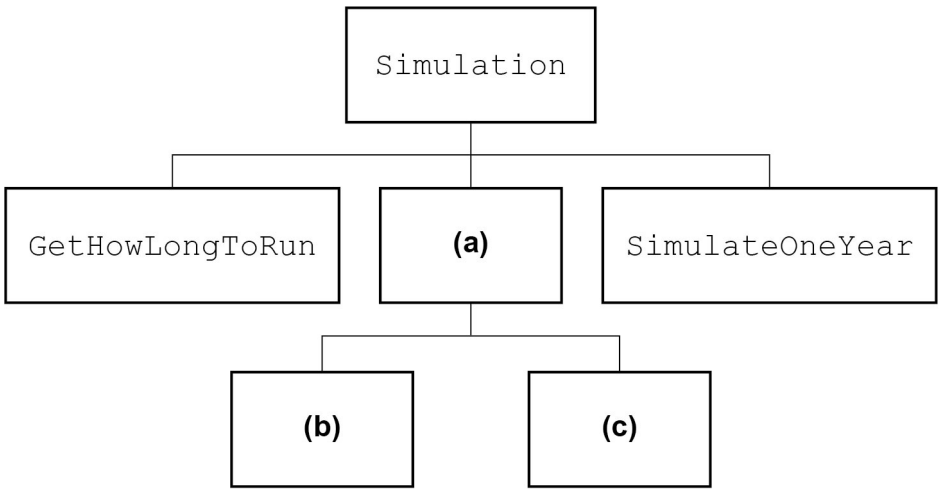
0 5 . 4 State the type of arithmetic operation carried out in the subroutine `CreateNewField` when the variables `Row` and `Column` are assigned new values outside the `FOR` loops.

State the values calculated for `Row` and `Column` using the values of `FIELDLENGTH` and `FIELDWIDTH` that are specified in the **Skeleton Program**.

[2 marks]

0 6 **Figure 6** shows an incomplete hierarchy chart for part of the **Skeleton Program**.

Figure 6



With reference to the **Skeleton Program** and **Figure 6**, answer the questions below.

0 6 . 1 What should be written in box **(a)** in **Figure 6**?

[1 mark]

0 6 . 2 What should be written in box **(b)** in **Figure 6**?

[1 mark]

0 6 . 3 What should be written in box **(c)** in **Figure 6**?

[1 mark]

0 6 . 4 Explain how data is shared between the separate subroutines.

[2 marks]

0 7 This question refers to the subroutine `ReadFile`.

0 7 . 1 Explain what will happen if the number of rows and columns supplied in the data file is greater than the number of rows and columns of the field modelled in the **Skeleton Program**.

[2 marks]

0 7 . 2 Assume it is a perfect growing year and there is no frost in spring and no drought in summer.

A **Data File** is read in which has an S (seed) in every possible location.

Explain what happens to the contents of `Field` in each season for the **first year only**.

[4 marks]

Turn over for Section C

Section C

You are advised to spend no more than **65 minutes** on this section.

Enter your answers to **Section C** in your Electronic Answer Document. You **must save** this document at regular intervals.

These questions require you to load the **Skeleton Program** and to make programming changes to it.

0 8 This question refers to the subroutine `GetHowLongToRun`.

What you need to do:

Task 1
Modify the subroutine `GetHowLongToRun` so that it checks that the value entered by the user is an integer between -1 to 5 inclusive.

If an invalid value, including a non-integer value, is entered for the duration of the simulation the program should output:

```
Invalid input
```

The subroutine should not return any values until a valid integer has been entered.

Task 2
Test that the changes you have made work by conducting the following test:

- start the simulation
- enter the value `-2`
- enter the value `6`
- enter the value `w`
- enter the value `1.4`
- enter the value `0` (zero).

Evidence that you need to provide
Include the following in your Electronic Answer Document.

0 8 . 1 Your amended PROGRAM SOURCE CODE for the subroutine `GetHowLongToRun`.

[5 marks]

0 8 . 2 SCREEN CAPTURE(S) showing the requested test.

[1 mark]

0 9 This question will extend the functionality of the simulation program.

An additional result is required from the simulation program. It is to calculate the percentage of `Field` cells occupied by plants.

The percentage is calculated by dividing the number of cells occupied by plants by the total number of cells and multiplying by 100.

What you need to do:

Task 1
Modify the subroutine `CountPlants` to calculate **and** display the percentage (rounded to the nearest whole number) of the field with plants.

Task 2
Test that your amended program code works by conducting the following test:

- start the simulation
- enter `1` (one year)
- enter `Y` (load a data file)
- enter the filename `TestCase.txt`

Evidence that you need to provide
Include the following in your Electronic Answer Document.

0 9 . 1 Your amended PROGRAM SOURCE CODE for the subroutine `CountPlants`.
[2 marks]

0 9 . 2 SCREEN CAPTURE(S) showing the `Field` contents **and** messages displayed for **spring**. Data for the other seasons should not be included in the evidence.
[1 mark]

Turn over for the next question

1 0 This question will extend the simulation program to allow `Field` data to be saved to a text file.

When the simulation has run for the required number of years, the user should be given the option of saving the `Field` data to a text file.

This text file must represent the data in the same format as the file `TestCase.txt` stores it (see **Figure 7**), including the numbers at the end of each line.

Figure 7

```
......S.............................|  0
....................X.............|  1
....................................|  2
.........SSSSSSXSSSSSSSSS...S.....|  3
....................................|  4
.........S.S...........S.S.........|  5
.........S.............S.........|  6
.........S.SSSSSSSSSSXSS.S.........|  7
.........S.S...........S.S.........|  8
..X.................................|  9
.........S.S.S.......S.S.S.........| 10
.........S.X.S.SSSSS.S.S.S.........| 11
.........S.S.S.S...S.S.S.S...X.....| 12
.........S.S.S.S.S.S.S.S.S.........| 13
.........S.S.S.S...S.S.S.S.........| 14
.........S.....SSSSS.....S.........| 15
.........S.S.S.......S.S.S.........| 16
..........XX........S..............| 17
.........S.S...........S.S.........| 18
.........S.............S.........| 19
```

What you need to do:

Task 1
Create a new subroutine `SaveToFile`. This subroutine is to ask the user whether they want to save the data by displaying the message:

```
Save the current Field state to a text file? (Y/N):
```

If the user chooses to save the data, the subroutine is to create a new text file with a user-specified filename and store the formatted data in it.

Task 2
In the subroutine `Simulation` add a call to the subroutine `SaveToFile` in an appropriate place.

Task 3
Test that the subroutine `SaveToFile` works by conducting the following test:

- run the simulation for **two** years
- do not load a data file
- save the data to file `Test1.txt`

Task 4
- run the simulation for **one** year
- load the data file `Test1.txt`

Evidence that you need to provide
Include the following in your Electronic Answer Document.

1 0 . 1 Your PROGRAM SOURCE CODE for the subroutines `SaveToFile` and `Simulation`.

[9 marks]

1 0 . 2 For the first run of the simulation: SCREEN CAPTURE(S) showing the `Field` displayed for **winter year 2** including the user interaction for saving the data.

[1 mark]

1 0 . 3 For the second run of the simulation: SCREEN CAPTURE(S) showing the `Field` displayed for **spring year 1** including the user interaction for loading the data file.

[1 mark]

Turn over for the next question

1 1 This question will further extend the functionality of the simulation program.

In the autumn, when plants drop their seeds, there may be a prevailing wind that shifts the seeds so they land in a different position.

If there is **no wind** the seeds land as shown in **Figure 8**.

Figure 8

```
.....
.SSS.
.SPS.
.SSS.
.....
```

If there is a prevailing wind from the **east** the seeds will land as shown in **Figure 9**.

Figure 9

```
.....
SSS..
S.P..  <----
SSS..
.....
```

If there is a prevailing wind from the **northwest** the seeds will land as shown in **Figure 10**.

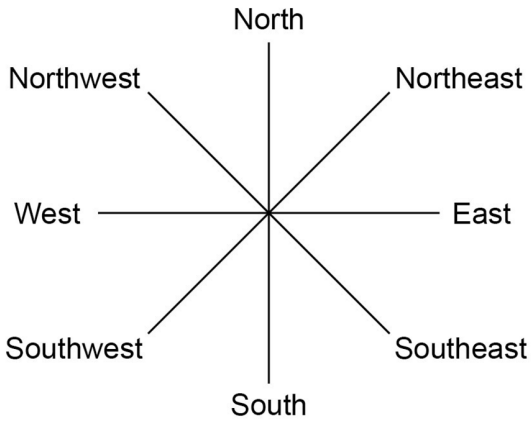
Figure 10

```
.....
.....
..PSS
..S.S
..SSS
```

The other wind directions as shown in **Figure 11** have similar effects on where the seeds land.

Each wind direction should occur randomly with each wind direction (or no wind at all) occurring with a 1 in 9 chance.

Figure 11



What you need to do:

Task 1
Modify the subroutine `SimulateAutumn` to change the position where the seeds land, depending on the prevailing wind direction.

Include code to output a message giving the name of the wind direction that was generated randomly. If there was no wind an appropriate message should be displayed.

Task 2
Test that your amended program code works by conducting the following test.

Without loading a data file run the simulation in stepping mode until **two** different wind directions have occurred.

Evidence that you need to provide
Include the following in your Electronic Answer Document.

1 1 . 1 Your amended PROGRAM SOURCE CODE for the subroutine `SimulateAutumn`.

[12 marks]

1 1 . 2 SCREEN CAPTURE(S) showing the `Field` contents **and** messages displayed for two autumns with different prevailing wind directions. Data for the other seasons does not need to be included in the evidence.

[1 mark]

END OF QUESTIONS

There are no questions printed on this page