AQA

AS
COMPUTER SCIENCE

Paper 1

Monday 4 June 2018 Morning Time allowed: 1 hour 45 minutes

Materials
For this paper you must have:
- 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 3 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 **Table 1** lists some terms that are important in the theory of computation. These terms are described in **Table 2**.

Table 1

Label	Term
A	Information hiding
B	Procedural abstraction
C	Functional abstraction
D	Data abstraction
E	Problem abstraction
F	Decomposition
G	Composition
H	Automation

Complete **Table 2** by filling in the unshaded cells with the appropriate labels **(A)** to **(H)** from **Table 1**. On each row write the label that most closely matches the description.

Table 2

Description of term	Label: A – H
Breaking a problem into a number of sub-problems	
Models are put into action to solve problems	
Combining procedures into compound procedures	
Details are removed until the problem is represented in a way that is possible to solve because the problem reduces to one that has already been solved	

Copy the contents of all the unshaded cells in **Table 2** into your Electronic Answer Document. **[4 marks]**

Turn over for the next question

0 2 The algorithm, represented using pseudo-code in **Figure 1**, describes a method to test whether an integer greater than 2 is prime or not.

Figure 1

```
INPUT Number
Root ← 1
WHILE (Root * Root) < Number
  Root ← Root + 1
ENDWHILE
d ← 2
FactorFound ← FALSE
WHILE (FactorFound = FALSE) AND (d <= Root)
  r ← Number MOD d
  IF r = 0 THEN
    FactorFound ← TRUE
  ENDIF
  d ← d + 1
ENDWHILE
IF FactorFound = FALSE THEN
  OUTPUT "Prime"
ELSE
  OUTPUT "Not prime"
ENDIF
```

The `MOD` operator calculates the remainder resulting from an integer division, for example, `10 MOD 3 = 1`.

0 2 . 1 Complete **Table 3** by hand-tracing the algorithm in **Figure 1**. Use `5` as the input value. You may not need to use all the rows in **Table 3**.

Table 3

Number	Root	d	FactorFound	r	Output

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

[3 marks]

0 2 . 2 Complete **Table 4** by hand-tracing the algorithm in **Figure 1**. Use `25` as the input value. You may not need to use all the rows in **Table 4**.

Table 4

Number	Root	d	FactorFound	r	Output

Copy the contents of all the unshaded cells in **Table 4** into your Electronic Answer Document.

[3 marks]

0 3 The algorithm, represented using pseudo-code in **Figure 2**, outputs a series of numbers. The contents of the series depends on the initial value entered by the user.

Figure 2

```
Number ← 0
WHILE (Number < 1) OR (Number > 10)
  OUTPUT "Enter a positive whole number: "
  INPUT Number
  IF Number > 10 THEN
    OUTPUT "Number too large."
  ELSE
    IF Number < 1 THEN
      OUTPUT "Not a positive number."
    ENDIF
  ENDIF
ENDWHILE
c ← 1
FOR k ← 0 TO Number - 1
  OUTPUT c
  c ← (c * (Number - 1 - k)) DIV (k + 1)
ENDFOR
```

The `DIV` operator calculates the result of an integer division, for example, `10 DIV 3 = 3`.

What you need to do:

Task 1
Write a program to implement the algorithm in **Figure 2**.

Task 2
Test the program by showing the result of entering:
`-3`
then `11`
then `10`

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

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

[9 marks]

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

[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 a single character.

[1 mark]

0 5 The program uses different variables to store single characters or the empty string. The variable identifiers are:
`FirstSignal`, `MenuOption`, `PlainTextLetter`, `Signal` and `Symbol`.

Using the variable identifiers listed above, complete **Table 5** by filling in the unshaded cells with the correct variable identifier.

Table 5

Variable identifier	Description
	uncoded letter, part of `PlainText`
	single unit of `Transmission` (= or `SPACE` or `EOL`)
	first character in `Transmission`
	used to build `SymbolString` (. or -)

Copy the contents of all the unshaded cells in **Table 5** into your Electronic Answer Document.

[4 marks]

0 6 Several subroutines use a variable `TransmissionLength`. It could have been declared as a global variable instead of locally in each subroutine.

Why is it good practice to use local variables?

[1 mark]

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

0 7 . 1 What is the purpose of the variable `i` in this subroutine?

[2 marks]

0 7 . 2 Explain what happens when the `GetTransmission` subroutine is called by the `ReceiveMorseCode` subroutine and the file to be read contains only spaces. You must include in your explanation how the subroutine `ReceiveMorseCode` deals with this situation.

[5 marks]

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

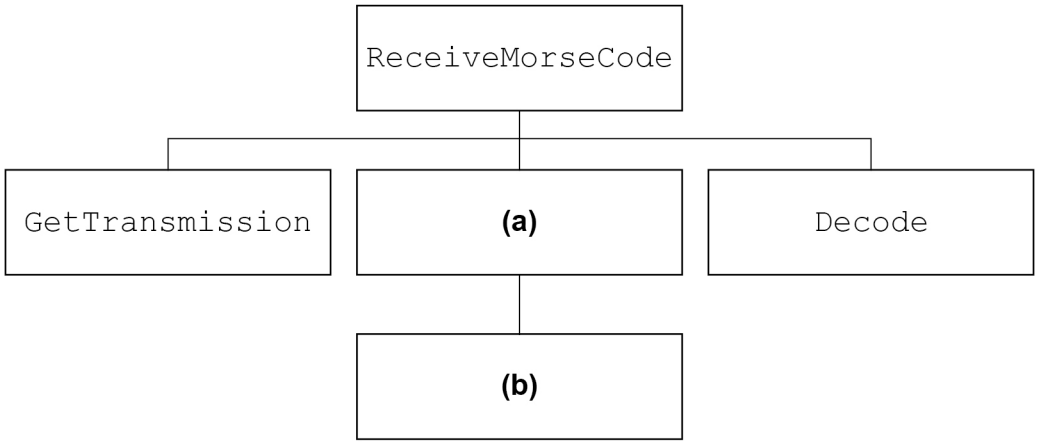
Give an example of a transmission string that would generate the error message `"Non-standard symbol received"` and explain how the code will detect the error caused by your string.

[3 marks]

0 9 **Figure 3** shows an incomplete hierarchy chart for part of the **Skeleton Program**.

With reference to the **Skeleton Program** and **Figure 3**, answer questions **9.1** to **9.2**.

Figure 3



0 9 . 1 What should be written in box **(a)** in **Figure 3**?

[1 mark]

0 9 . 2 What should be written in box **(b)** in **Figure 3**?

[1 mark]

Turn over for the next question

1 0 Morse codes also exist for numerals, so that a text message including numbers can also be transmitted. For example, the Morse code for 3 is:

•••−−

The program is to be extended to include the Morse code for all numerals from 0 to 9.

Describe the changes that would need to be made to the **Skeleton Program** to achieve this.

In your answer you should discuss the changes that need to be made to the data structures and subroutines.

You are **not** expected to actually make the changes.

[6 marks]

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.

1 1 This question refers to the subroutine `SendMorseCode`.

What you need to do:

Task 1
Modify the subroutine `SendMorseCode` so that it checks each character of the input string entered by the user.

If a character is not an uppercase letter or a space then the program should use the subroutine `ReportError` to output a suitable error message.

The subroutine should then set `MorseCodeString` to the empty string and exit.

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

- run the program
- choose option `S`
- enter the string `Help`

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

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

[4 marks]

1 1 . 2 SCREEN CAPTURE(S) showing the requested test.

[1 mark]

Turn over for the next question

1 2 This question will extend the functionality of the program. To send a Morse code message, the dots and dashes need to be converted into signals.

What you need to do:

Task 1
Create a new subroutine `SendSignals` that converts a Morse code message such as

```
-⌂.⌂.-⌂⌂⌂-..-
```

(`TEA X`) to its equivalent signal

```
===⌂⌂⌂=⌂⌂⌂=⌂===⌂⌂⌂⌂⌂⌂⌂===⌂=⌂=⌂===⌂
```

by using the rules

- a dot is converted to an equals sign (=) followed by a space
- a dash is converted to three equals signs (===) followed by a space
- a space is converted to two spaces.
  (This results in three spaces after the signals for a letter.)

Note that ⌂ represents a space and should be displayed as a space on screen.

The subroutine is to output the complete transmission string to the screen.

Add the call `SendSignals(MorseCodeString)` as the last statement to be executed in the subroutine `SendMorseCode`.

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

- run the program
- choose option `S`
- enter the string `MORSE X`

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

1 2 . 1 Your PROGRAM SOURCE CODE for the subroutine `SendSignals`.
[7 marks]

1 2 . 2 SCREEN CAPTURE(S) showing the Morse code message and the signals.
[1 mark]

1 3 This question will extend the program to output the letters of the alphabet and their corresponding Morse codes.

What you need to do:

Task 1
Create a new subroutine `OutputAlphabetWithCode`. This subroutine is to display each uppercase letter of the alphabet and its corresponding Morse code. The output is to be created using the data in `Letter` and `MorseCode`.

The format of the output should be as shown in **Figure 4**.

- Each uppercase letter is followed by a space and then its Morse code, grouped four to a line.
- Each Morse code should be padded with trailing spaces to a width of 6 characters.

Figure 4

```
A .-    B -...  C -.-.  D -..
E .     F ..-.  G --.   H ....
I ..    J .---  K -.-   L .-..
M --    N -.    O ---   P .--.
Q --.-  R .-.   S ...   T -
U ..-   V ...-  W .--   X -..-
Y -.--  Z --..
```

Task 2
Amend the subroutines `DisplayMenu` and `SendReceiveMessages` so that the user can choose a new option `A` that will call `OutputAlphabetWithCode`.

Task 3
- run the program
- choose new option `A`

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

1 3 . 1 Your PROGRAM SOURCE CODE for the subroutine `OutputAlphabetWithCode`. Your amended PROGRAM SOURCE CODE for the subroutines `DisplayMenu` and `SendReceiveMessages`.

[6 marks]

1 3 . 2 SCREEN CAPTURE(S) showing the main menu and the output from the test run.

[1 mark]

Turn over for the next question

1 4 This question will further extend the functionality of the program.

Before sending a message in Morse code the message is to be encrypted using a Caesar cipher. To encrypt a character (letter or space) it is shifted a given number of places (the key) in the alphabet (space is taken as the first character, then A, B, C and so on to Z). If a character is required before space or after Z then wrap-round is to be implemented.

For example:

- if the key is 2, then SPACE is encrypted as B, A is encrypted as C, B is encrypted as D, …, X is encrypted as Z, Y is encrypted as SPACE, and Z is encrypted as A.
- if the key is -1, then SPACE is encrypted as Z, A is encrypted as SPACE, B is encrypted as A, …, and Z is encrypted as Y.

Three different keys are to be used depending on the position of the character in the message.

The first character in the message is to be encrypted using the first key, the second character is encrypted with the second key and the third character with the third key.

The fourth character is then encrypted using the first key again, the fifth character is encrypted with the second key and the sixth character with the third key and so on.

The three keys are repeatedly used like this until the end of the message.

What you need to do:

Task 1
At the start of the subroutine `SendReceiveMessages` add code to enter three values that **must** be integers (the keys).

Task 2
Amend the subroutine `SendMorseCode` to encrypt the message using the substitution method described above and the keys entered by the user.

Task 3
Test that your code works by conducting the following test:

- run the program
- for the first key enter `17`
- for the second key enter `5`
- for the third key enter `-3`
- enter `S`
- enter the string `TEA X`

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

1 4 . 1 Your amended PROGRAM SOURCE CODE for the subroutines `SendMorseCode` and `SendReceiveMessages`.

[9 marks]

1 4 . 2 SCREEN CAPTURE(S) showing the keys entered, the plain text and the encrypted Morse code message.

[1 mark]

END OF QUESTIONS

There are no questions printed on this page.

JUN187516 1