AQA

A-level
COMPUTER SCIENCE

Paper 1

Monday 11 June 2018 Morning Time allowed: 2 hours 30 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 and a hard copy of the Preliminary Material
- an electronic version of the Data File **aqawords.txt**.

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 (also at the top of 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 100.
- No extra time is allowed for printing and collating.
- The question paper is divided into **four** sections.

Advice
You are advised to allocate time to each section as follows:
Section A – 45 minutes; **Section B** – 20 minutes; **Section C** – 15 minutes; **Section D** – 70 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.

There are no questions printed on this page

Section A

You are advised to spend no longer than **45 minutes** on this section.

Type your answers to **Section A** in your Electronic Answer Document.

You **must save** this document at regular intervals.

0 1 There are three boxes containing vegetables. One contains onions, one contains carrots and one contains onions and carrots. The three boxes have been labelled. One is labelled "onions", one is labelled "carrots" and the other is labelled "onions and carrots". You know that all three have been labelled incorrectly.

0 1 . 1 Describe how you can work out what each box actually contains by taking just **one** vegetable out of **one** box, without looking inside any of the boxes.

[2 marks]

0 2 . 1 How would the infix expression `5 - 3` be represented in Reverse Polish notation?

[1 mark]

0 2 . 2 How would the infix expression `3 + 4 * 2 - 1` be represented in Reverse Polish notation?

Use the space below for rough working then copy your answer into your Electronic Answer Document.

[2 marks]

Question 02 continues on the next page

0 2 . 3 Explain why Reverse Polish notation is sometimes used instead of infix notation.

[2 marks]

0 2 . 4 To evaluate an expression in Reverse Polish notation, you start from the left hand side of the expression and look at each item until you find an operator (eg + or −).

This operator is then applied to the two values immediately preceding it in the expression. The result obtained from this process replaces the operator and the two values used to calculate it. This process continues until there is only one value in the expression, which is the final result of the evaluation.

For example `5 2 7 + +` would change to `5 9 +` after the first replacement.

Explain how a stack could be used in the process of evaluating an expression in Reverse Polish notation.

[3 marks]

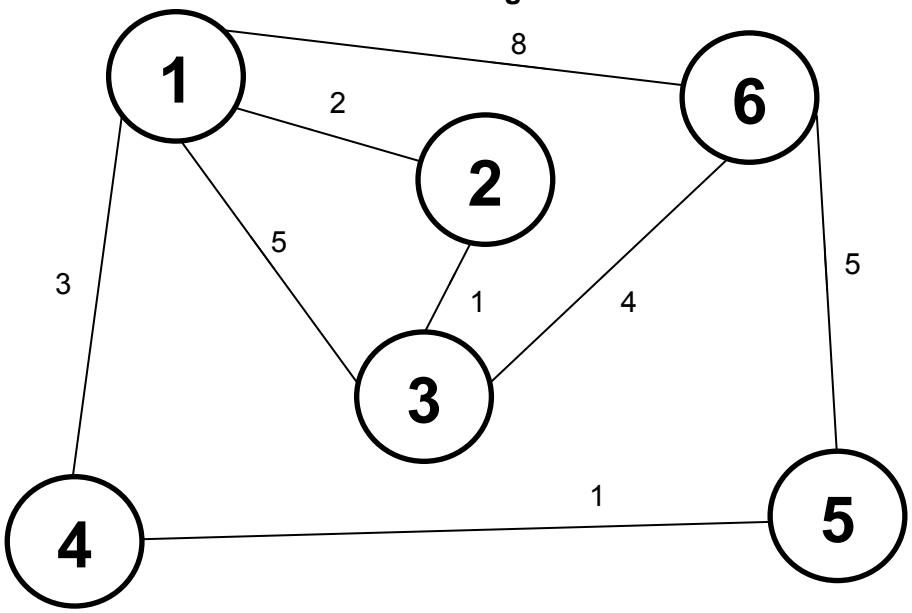
0 2 . 5 Stacks are also used to store a stack frame each time a subroutine call is made. State **two** components of a stack frame.

[2 marks]

There are no questions printed on this page

0 3 **Figure 1** is a graph that shows the time it takes to travel between six locations in a warehouse. The six locations have been labelled with the numbers 1 - 6. When there is no edge between two nodes in the graph this means that it is not possible to travel directly between those two locations. When there is an edge between two nodes in the graph the edge is labelled with the time (in minutes) it takes to travel between the two locations represented by the nodes.

Figure 1



0 3 . 1 The graph is represented using an adjacency matrix, with the value 0 being used to indicate that there is no edge between two nodes in the graph.

A value should be written in every cell.

Complete the unshaded cells in **Table 1** so that it shows the adjacency matrix for **Figure 1**.

Table 1

	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

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

[2 marks]

0 3 . 2 Instead of using an adjacency matrix, an adjacency list could be used to represent the graph. Explain the circumstances in which it would be more appropriate to use an adjacency list instead of an adjacency matrix.

[2 marks]

0 3 . 3 State **one** reason why the graph shown in **Figure 1** is **not** a tree.

[1 mark]

0 3 . 4 The graph in **Figure 1** is a weighted graph. Explain what is meant by a **weighted graph**.

[1 mark]

Question 03 continues on the next page

Figure 2 contains pseudo-code for a version of Djikstra's algorithm used with the graph in **Figure 1**.

Q is a priority queue which stores nodes from the graph, maintained in an order based on the values in array D. The reordering of Q is performed automatically when a value in D is changed.

AM is the name given to the adjacency matrix for the graph represented in **Figure 1.**

Figure 2

```
Q ← empty queue
FOR C1 ← 1 TO 6
  D[C1] ← 20
  P[C1] ← -1
  ADD C1 TO Q
ENDFOR
D[1] ← 0
WHILE Q NOT EMPTY
  U ← get next node from Q
  remove U from Q
  FOR EACH V IN Q WHERE AM[U, V] > 0
    A ← D[U] + AM[U, V]
    IF A < D[V] THEN
      D[V] ← A
      P[V] ← U
    ENDIF
  ENDFOR
ENDFOR
OUTPUT D[6]
```

0 3 . 5 Complete the unshaded cells of **Table 2** to show the result of tracing the algorithm shown in **Figure 2**. Some of the trace, including the maintenance of Q, has already been completed for you.

[7 marks]

Table 2

				D						P					
U	Q	V	A	1	2	3	4	5	6	1	2	3	4	5	6
-	1,2,3,4,5,6	-	-	20	20	20	20	20	20	-1	-1	-1	-1	-1	-1
				0											
1	2,3,4,5,6	2													
		3													
		4													
		6													
2	3,4,5,6	3													
3	4,5,6	6													
4	5,6	5													
5	6	6													
6	-														

Copy the contents of the unshaded cells in **Table 2** into the table in your Electronic Answer Document.

0 3 . 6 What does the output from the algorithm in **Figure 2** represent?

[1 mark]

0 3 . 7 The contents of the array `P` were changed by the algorithm. What is the purpose of the array `P`?

[2 marks]

Turn over for the next question

0 4 . 1 Describe the Halting problem.

[2 marks]

0 4 . 2 Why is it not possible to create a Turing machine that solves the Halting problem?

[1 mark]

0 4 . 3 To define a Turing machine the finite alphabet of symbols that it can use needs to be specified and there needs to be a tape.

State **two** other components of a Turing machine.

[2 marks]

0 4 . 4 Explain what a Universal Turing Machine is.

[2 marks]

0 4 . 5 Why can a Universal Turing Machine be considered to be more powerful than any computer that you can purchase?

[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.

The question 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 5 Write a program that checks which numbers from a series of numbers entered by the user are prime numbers.

The program should get a number from the user and then display the messages:

- `"Not greater than 1"` if the number entered is 1 or less
- `"Is prime"` if the number entered is a prime number
- `"Is not prime"` otherwise.

The user should then be asked if they want to enter another number and the program should repeat if they say that they do.

A prime number is a positive integer that will leave a remainder if it is divided by any positive integer other than 1 and itself.

You may assume that each number entered by the user is an integer.

If your program only works correctly for some prime numbers you will get some marks for this question. To get full marks for this question, your program must work correctly for any valid integer value that the user enters.

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

0 5 . 1 Your PROGRAM SOURCE CODE.

[12 marks]

0 5 . 2 SCREEN CAPTURE(S) showing the result of testing the program by:

- entering the number `1`
- then choosing to enter another number
- then entering the number `5`
- then choosing to enter another number
- then entering the number `8`
- and then choosing not to enter another number.

[1 mark]

Section C

You are advised to spend no more than **15 minutes** on this section.

Type your answers to **Section C** into 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 6 . 1 State the name of an identifier for a built-in function used in the Skeleton Program that has a string parameter and returns an integer value.

[1 mark]

0 6 . 2 State the name of an identifier for a local variable in a method in the `QueueOfTiles` class.

[1 mark]

0 6 . 3 The `QueueOfTiles` class implements a linear queue. A circular queue could have been used instead.

Explain why a circular queue is often a better choice of data structure than a linear queue.

[2 marks]

0 6 . 4 It could be argued that the algorithms for a linear queue lead to simpler program code.

State **one** other reason why a linear queue is an appropriate choice in the Skeleton Program even though circular queues are normally a better choice.

[1 mark]

0 6 . 5 State **one** additional attribute that must be added to the `QueueOfTiles` class if it were to be implemented as a circular queue instead of as a linear queue.

[1 mark]

0 6 . 6 Describe the changes that would need to be made to the Skeleton Program so that the probability of a player getting a one point tile is the same as the probability of them getting a tile worth more than one point. The changes you describe should not result in any changes being made to the points value of any tile.

You should **not** change the Skeleton Program when answering this question.

[4 marks]

0 6 . 7 The `GetChoice` subroutine uses a built-in function to convert a string to uppercase.

Describe how a string consisting of lowercase characters could be converted to uppercase using only one iterative structure if the programming language being used does not have a built-in function that can do this conversion.

You may assume that only lowercase characters are entered by the user.

You should not change the Skeleton Program when answering this question.

[4 marks]

Turn over for Section D

Section D

You are advised to spend no more than **70 minutes** on this section.

Type your answers to **Section D** 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 7 This question refers to the subroutine `CreateTileDictionary`.

The points values for the letters J and X are to be changed so that they are not worth as many points as the letters Z and Q.

Adapt the subroutine `CreateTileDictionary` so that the letters J and X are worth 4 points instead of 5 points.

Test that the changes you have made work:

- run the **Skeleton Program**
- enter `2` at the main menu
- enter `1` to view the letter values.

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

0 7 . 1 Your PROGRAM SOURCE CODE for the amended subroutine `CreateTileDictionary`.

[1 mark]

0 7 . 2 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

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

Currently each player starts with 15 letter tiles. In the `Main` subroutine `StartHandSize` is set to a value of 15.

Change the subroutine `Main` so that the user can choose what the value for `StartHandSize` will be.

Before the main menu is displayed and before the first iteration structure in `Main` the message `"Enter start hand size: "` should be displayed and the user's input should be stored in `StartHandSize`. This should happen repeatedly until the user enters a value for the start hand size that is between 1 and 20 inclusive.

Test that the changes you have made work:

- run the **Skeleton Program**
- enter `0` when asked to enter the start hand size
- enter `21` when asked to enter the start hand size
- enter `5` when asked to enter the start hand size
- enter `1` at the main menu.

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

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

[4 marks]

0 8 . 2 SCREEN CAPTURE(S) showing the requested test. You must make sure that evidence for all parts of the requested test is provided in the SCREEN CAPTURE(S).

[1 mark]

Turn over for the next question

0 9 This question refers to the subroutine `CheckWordIsValid`.

When a player enters a word, a linear search algorithm is used to check to see if the word entered is in the list of `AllowedWords`. The subroutine `CheckWordIsValid` is to be changed so that it uses a more time-efficient search algorithm.

Change the `CheckWordIsValid` subroutine so that it uses a binary search algorithm instead of a linear search algorithm.

You **must write your own search routine** and not use any built-in search function that might be available in the programming language you are using.

Each item in `AllowedWords` that is compared to the word that the user entered should be displayed on the screen.

Figure 4 shows examples of how the new version of `CheckWordIsValid` should work if `AllowedWords` contained the items shown in **Figure 3**.

Figure 3

BIG
BUG
FED
GET
JET
NOT
SIP
WON

Figure 4

1) If the user enters the word BIG then a value of `True` should be returned and the words GET, BUG, BIG should be displayed, in that order.
2) If the user enters the word JET then a value of `True` should be returned and the words GET, NOT, JET should be displayed, in that order.
3) If the user enters the word ZOO then a value of `False` should be returned and the words GET, NOT, SIP, WON should be displayed, in that order.

Test that the changes you have made work:

- run the **Skeleton Program**
- if you have answered **Question 8**, enter `15` when asked to enter the start hand size; if you have not answered **Question 8** yet then skip this step
- enter `2` at the main menu
- enter the word `jars`.

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

0 9 . 1 Your PROGRAM SOURCE CODE for the amended subroutine `CheckWordIsValid`.

[8 marks]

0 9 . 2 SCREEN CAPTURE(S) showing the requested test.

[1 mark]

Turn over for the next question

1 0 This question extends the functionality of the game.

After spelling a valid word the player decides which one of four options to select to determine how many tiles will be added to their hand. Before choosing they can look at the values of the tiles.

It would help the player make their decision if they were aware of how useful each letter was by knowing the frequency with which each letter appears in the list of allowed words.

The program is to be extended so that when the player chooses to view the tile values they are also shown the number of times that each letter appears in the list of allowed words.

What you need to do

Task 1
Create a new subroutine called `CalculateFrequencies` that looks through the list of allowed words and displays each of the 26 letters in the alphabet along with the number of times that the letter appears in the list of allowed words, which the subroutine has calculated.

Task 2
Modify the `DisplayTileValues` subroutine so that after displaying the tile values it also calls the `CalculateFrequencies` subroutine.

Task 3
Test that the changes you have made work:

- run the **Skeleton Program**
- if you have answered **Question 8**, enter `15` when asked to enter the start hand size; if you have not answered **Question 8** yet then skip this step
- enter `2` at the main menu
- enter `1` to display the letter values.

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

1 0 . 1 Your PROGRAM SOURCE CODE for the new subroutine `CalculateFrequencies`, the amended subroutine `DisplayTileValues` and any other subroutines you have modified when answering this question.

[8 marks]

1 0 . 2 SCREEN CAPTURE(S) showing the requested test.

[1 mark]

1 1 The scoring system for the game is to be changed so that if a player spells a valid word they score points for all valid words that are a prefix of the word entered. A prefix is the first **x** characters of a word, where **x** is a whole number between one and the number of characters in the word.

In the **Skeleton Program**, `AllowedWords` contains the list of valid words that have been read in from the **Data File aqawords.txt**.

Example
If the user enters the word TOO they will be awarded points for the valid words TOO and TO as TO is a prefix of the word TOO. They will not be awarded points for the word OO even though it is a valid word and is a substring of TOO because it is not a prefix of TOO.

Example
If the user enters the word BETTER they will be awarded points for the words BETTER, BET and BE as these are all valid prefixes of the word entered by the user. They would not be awarded points for BETT or BETTE as these are not valid English words. They would not be awarded points for BEER as even though it is contained in the word BETTER it is not a prefix.

Example
If the user enters the word BIOGASSES they will be awarded points for the words BIOGASSES, BIOGAS, BIOG, BIO and BI as these are all valid prefixes of the word entered by the user. They would not be awarded points for BIOGA, BIOGASS or BIOGASSE as these are not valid English words. They would not be awarded points for GAS as even though it is contained in the word BIOGASSES it is not a prefix.

Example
If the user enters the word CALMIEST they will not be awarded any points as even though CALM at the start is a valid word the original word entered by the user, CALMIEST, is not.

Example
If the user enters the word AN they will be awarded points for the word AN. They would not be awarded points for A even though A at the start is a valid word as points are only awarded for words that are at least two letters long.

What you need to do

Task 1
Write a **recursive** subroutine called `GetScoreForWordAndPrefix` that, if given a valid word, returns the score of the word added to the score for any valid words that are prefixes of the word.

To get full marks for this task the `GetScoreForWordAndPrefix` subroutine must make use of recursion in an appropriate way to calculate the score for any prefixes that are also valid words.

If your solution uses an alternative method to recursion you will be able to get most but not all of the available marks for this question.

Question 11 continues on the next page

Task 2
Modify the `UpdateAfterAllowedWord` subroutine so that it calls the new `GetScoreForWordAndPrefix` subroutine instead of the `GetScoreForWord` subroutine.

Task 3
Test that the changes you have made to the program work:

- run the **Skeleton Program**
- if you have answered **Question 8**, enter `15` when asked to enter the start hand size; if you have not answered **Question 8** yet then skip this step
- enter `2` at the main menu
- enter the word `abandon`
- enter `4` so that no tiles are replaced.

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

1 1 . 1 Your PROGRAM SOURCE CODE for the new subroutine `GetScoreForWordAndPrefix`, the amended subroutine `UpdateAfterAllowedWord` and any other subroutines you have modified when answering this question.

[11 marks]

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

[1 mark]

END OF QUESTIONS