AQA

General Certificate of Education
Advanced Subsidiary Examination
June 2015

Computing COMP1

Unit 1 Problem Solving, Programming, Data Representation and Practical Exercise

Monday 1 June 2015 9.00 am to 11.00 am

You will need to:
- access the Electronic Answer Document
- refer to the Preliminary Material and Skeleton Program.

You must **not** use a calculator.

Time allowed
- 2 hours

Instructions
- Type the information required on the front of your Electronic Answer Document.
- Enter your answers into the Electronic Answer Document.
- Answer **all** questions.
- You will need access to:
  - a computer
  - a printer
  - appropriate software
  - an electronic version of the Skeleton Program
  - a hard copy of the Preliminary Material.
- 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 of your Electronic Answer Document (not the front cover).

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.
- You are advised to spend time on each section as follows:
  Section A – 35 minutes
  Section B – 15 minutes
  Section C – 10 minutes
  Section D – 60 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 unit if your details are not on every page of your Electronic Answer Document.

Section A

You are advised to spend no more than **35 minutes** on this section.

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

Question 1

0 1 Represent the **denary** number 55 using **8-bit unsigned binary**.

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

[1 mark]

0 2 Represent the **denary** number 55 using **hexadecimal**.

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

[1 mark]

0 3 Why are bit patterns often displayed using hexadecimal instead of binary?

[1 mark]

0 4 Represent the denary number -59 as an **8-bit two's complement binary integer**.

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

[2 marks]

0 5 Represent the denary number 5.625 as an **unsigned binary fixed point number** with three bits before and five bits after the binary point.

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

[2 marks]

The ASCII system uses 7 bits to represent a character. The ASCII code in denary for the numeric character ‘0’ is 48; other numeric characters follow on from this in sequence. The numeric character ‘0’ is represented using 7 bits as `0110000`.

0 6 Using 7 bits, express the ASCII code for the character '6' in binary.

[1 mark]

0 7 How many different character codes can be represented using 7-bit ASCII?

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

[1 mark]

Examples of logical bitwise operators include `AND`, `OR`, `NOT` and `XOR`.

0 8 Describe how **one** of these logical bitwise operators can be used to convert the 7-bit ASCII code for a numeric character into a 7-bit pure binary representation of the number, (eg `0110001` for the numeric character '1' would be converted to `0000001`).

[2 marks]

Turn over for the next question

Characters are transmitted using an 8-bit code that includes the 7-bit ASCII code and a single parity bit in the most significant bit. A parity bit is added for error checking during data transmission.

0 9 Using even parity, what 8-bit code is sent for the numeric character '0'?

[1 mark]

Hamming code is an alternative to the use of a single parity bit. Hamming code uses multiple parity bits - this allows it to correct some errors that can occur during transmission.

The parity bits are located in the power of two bit positions (1, 2, 4, 8, etc.). The other bit positions are used for the data bits.

1 0 Describe how the receiver can detect and correct a single-bit error using Hamming code.

[4 marks]

Figure 1 shows the bit pattern **received** in a communication that is using **even parity** Hamming code. The data bits received are `1101000`.

Figure 1

Bit position	11	10	9	8	7	6	5	4	3	2	1
Bit	1	1	0	0	1	0	0	0	0	0	1

There has been a single-bit error in the data transmission.

1 1 Which bit position from the bit pattern in **Figure 1** contains an error?

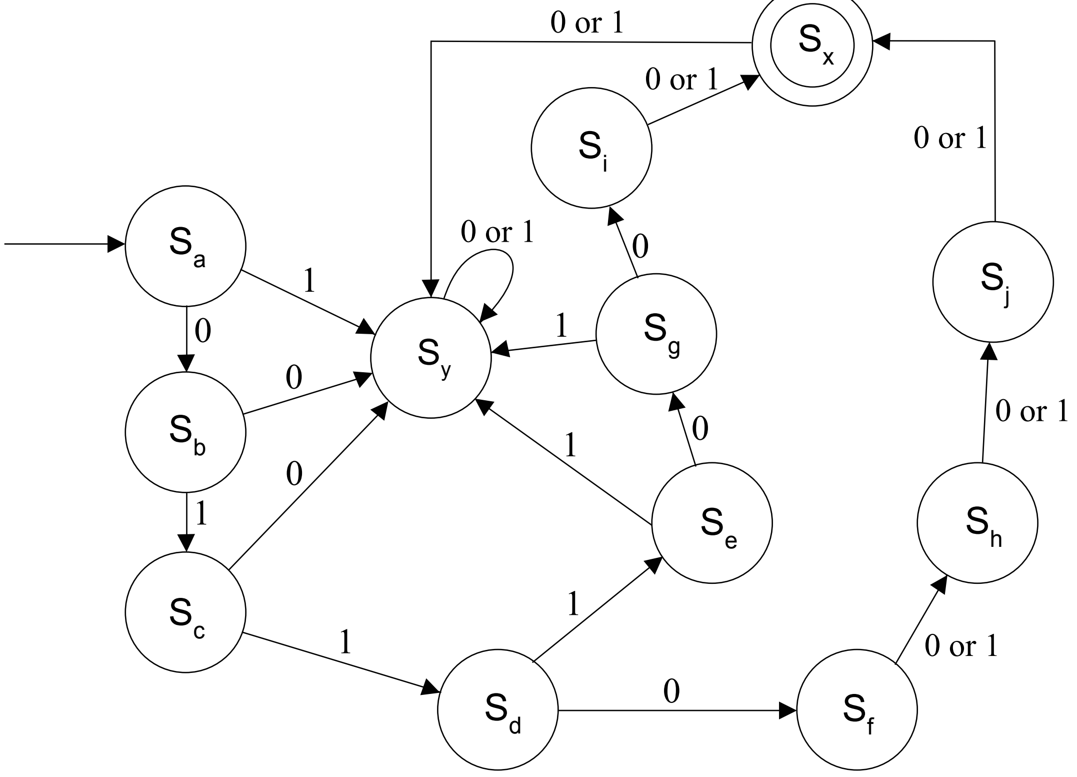
Use the space below for rough working, then copy the answer to your Electronic Answer Document.

[1 mark]

Figure 2 shows a Finite State Machine (FSM) represented as a state transition diagram.

The machine takes a bit pattern as an input. The bit pattern is considered to be valid if the machine ends up in the accepting state S_x. The bit patterns `0111000` and `0110001` are valid; the bit patterns `010011` and `01110111` are not valid. Bit patterns are processed starting with the left-most bit.

Figure 2



1 2 Is the bit pattern `0111010` a valid input for the FSM shown in **Figure 2**?

[1 mark]

1 3 A finite state machine can also be represented as a state transition table. **Table 1** shows part of the state transition table that represents the finite state machine shown in **Figure 2**. The state transition table is only partially complete.

Table 1

Initial State	Input	New State
S_g	1	
S_y		
S_y		

Complete **Table 1** by filling in the unshaded cells.

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

[2 marks]

Turn over ▸

1 4 What is the purpose of the FSM shown in **Figure 2**?

[1 mark]

1 5 The state S_i in the FSM shown in **Figure 2** is not necessary and is going to be removed from the FSM.

Describe the change that needs to be made so that state S_i can be removed without changing the functionality of the FSM.

[1 mark]

Question 2 Images are often represented in a computer's memory using vector graphics. A vector graphic consists of a collection of objects.

1 6 State **three** items of data that would need to be stored about a circle object if it is to be represented using vector graphics.

[3 marks]

Instead of representing an image as a vector graphic, it could be represented as a bitmapped image.

1 7 Describe how an image can be represented as a bitmapped image in a computer's memory.

[3 marks]

1 8 Describe **three** advantages of using vector graphics instead of bitmaps to represent images.

[3 marks]

Question 3 A pseudo-code representation of an algorithm is given in **Figure 3**.

The function `Len` takes a string, `S`, and returns an integer value that is the number of characters in the string `S`, eg

`Len("Hello")` would return a value of `5`

The function `GetChar` takes a string, `S`, and an integer value, `n`, and returns the character that is in the `n`th position in the string `S`, eg

`GetChar("Hello", 2)` would return the character `e`
`GetChar("Hello", 1)` would return the character `H`

The function `Concat` takes two strings, `S1` and `S2`, and returns a string that is the string `S1` followed by the string `S2`, eg

`Concat("Hello"," sunshine")` would return the string
`Hello sunshine`

Figure 3

```
INPUT IStr
OStr ← ""
Count ← 1
WHILE Count <= Len(IStr) DO
  OStr ← Concat(GetChar(IStr, Count), OStr)
  Count ← Count + 1
ENDWHILE
```

1 9 Dry run the algorithm shown in **Figure 3** by completing **Table 2** for when the user enters the string "`Lou`". The first row has been done for you.

Copy the cells in **Table 2** that contain your answer into the Electronic Answer Document.

Table 2

IStr	**OStr**	**Count**
Lou	▨	1

[5 marks]

Turn over ▸

Section B

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

Type 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.**

- Save your program/project/file in its own folder/directory.
- You are advised to save your program at regular intervals.

Question 4 Create a folder/directory **Question4** for your new program.

The algorithm, represented using pseudo-code in **Figure 4**, and the variable table, **Table 3**, describe a program that calculates and displays all of the prime numbers between 2 and 50, inclusive.

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

If you are unsure how to use the `MOD` operator in the programming language you are using, there are examples of it being used in the **Skeleton Program**.

Figure 4

```
OUTPUT "The first few prime numbers are:"
FOR Count1 ← 2 TO 50 DO
  Count2 ← 2
  Prime ← "Yes"
  WHILE Count2 * Count2 <= Count1 DO
    IF (Count1 MOD Count2 = 0) THEN
      Prime ← "No"
    ENDIF
    Count2 ← Count2 + 1
  ENDWHILE
  IF Prime = "Yes" THEN
    OUTPUT Count1
  ENDIF
ENDFOR
```

Table 3

Identifier	**Data Type**	**Purpose**
`Count1`	`Integer`	Stores the number currently being checked for primeness
`Count2`	`Integer`	Stores a number that is being checked to see if it is a factor of `Count1`
`Prime`	`String`	Indicates if the value stored in `Count1` is a prime number or not

What you need to do

Write a program for the algorithm in **Figure 4**.

Run the program and test that it works correctly.

Save the program in your new **Question4** folder/directory.

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

2 0 Your PROGRAM SOURCE CODE.

[11 marks]

2 1 SCREEN CAPTURE(S) for the test showing the correct working of the program.

[1 mark]

2 2 Describe the changes that would need to be made to the algorithm shown in **Figure 4**, so that instead of displaying the prime numbers between 2 and 50, inclusive, it displays all the prime numbers between 2 and a value input by the user, inclusive.

[3 marks]

Turn over for Section C

Turn over ▶

Section C

You are advised to spend no more than **10 minutes** on this section.

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

These questions refer to the **Preliminary Material** and require you to load 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.

Question 5 State the name of an identifier for:

2 3 a named constant

[1 mark]

2 4 a user-defined subroutine that has only one parameter

[1 mark]

2 5 a two-dimensional array or an equivalent list.

[1 mark]

Look at the `MakeMove` subroutine.

2 6 Describe how this subroutine could be rewritten so that instead of there being three lines of code that change the value in the start square, there could be just one line of code that does this. The functionality of the `MakeMove` subroutine should not be altered by the changes you describe.

[1 mark]

Look at the last selection structure in the MAIN PROGRAM BLOCK.

2 7 What is the purpose of this selection structure?

[2 marks]

2 8 The logic of a selection structure can be represented using a decision table. **Table 4** shows an attempt to represent the logic of this selection structure.

Table 4

Conditions	`>= 97`	Y	N	Y	Y
	`<= 123`	N	Y	N	Y
Action	Change value of `PlayAgain`		X	X	X

Explain why this decision table is not an accurate representation of the logic of this selection structure.

[3 marks]

Turn over for Section D

Section D

You are advised to spend no more than **60 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 make programming changes to it.

Question 6 This question refers to the MAIN PROGRAM BLOCK and will extend the functionality of the **Skeleton Program**.

The number of moves made by the players in a game of CAPTURE THE SARRUM will be tracked. A variable called `NoOfMoves` will be used to store the number of moves completed by the players. At the start of every game `NoOfMoves` should be set to an initial value of zero.

Each time a player enters a legal move `1` should be added to the value stored in `NoOfMoves`.

After the call to the `MakeMove` subroutine, the `NoOfMoves` variable should be updated and then a message should be displayed saying "`The number of moves completed so far: n`" – where **n** is the value of `NoOfMoves`.

Test that the changes you have made work:

- run the **Skeleton Program**
- enter `N` when asked if you want to play the sample game
- enter a start square of `17` and a finish square of `16`
- enter a start square of `12` and a finish square of `13`
- enter a start square of `18` and a finish square of `17`.

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

2 9 Your PROGRAM SOURCE CODE for the amended MAIN PROGRAM BLOCK.
[4 marks]

3 0 SCREEN CAPTURE(S) showing the requested test.
[2 marks]

Question 7 This question refers to the subroutine `CheckMoveIsLegal`.

When the user has entered the start square and the finish square for their move, a number of checks are made to see if their intended move is legal.

Add a validation check to the subroutine `CheckMoveIsLegal`, so that a move is accepted as being legal only if the **finish square** refers to a square that is on the board.

Test that the changes you have made work:

- run the **Skeleton Program**
- enter `N` when asked if you want to play the sample game
- enter a start square of `88` and a finish square of `98`
- enter a start square of `18` and a finish square of `19`
- enter a start square of `87` and a finish square of `86`.

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

3 1 Your PROGRAM SOURCE CODE for the amended subroutine `CheckMoveIsLegal`.

[4 marks]

3 2 SCREEN CAPTURE(S) showing the requested test. You must make sure that evidence for all parts of the requested test by the user is provided in the SCREEN CAPTURE(S).

[3 marks]

Turn over for Question 8

Turn over ▶

Question 8 This question will extend the functionality of the game.

A new type of piece is to be added to the game – a kashshaptu (witch). When a redum is promoted, instead of changing into a marzaz pani it changes into a kashshaptu.

A kashshaptu moves in the same way as a sarrum (one square in any direction). Like all the other pieces, a kashshaptu cannot move into a square containing a piece of the same colour and when a kashshaptu is moved into a square containing the opponent's sarrum, it captures the sarrum and the game is over.

When a move would cause the kashshaptu to enter a square containing one of the opponent's pieces (except the sarrum), the kashshaptu stays in its start square and the opponent's piece changes colour and now belongs to the player who played the kashshaptu. It then becomes the other player's turn.

Figures 5 and **6** show diagrams giving examples of legal kashshaptu moves. A kashshaptu is represented by the symbol `K`.

Figure 5

1								
2	▨	▨	▨					
3	▨	WK	▨				WG	
4	▨	▨	▨					
5								
6		WS					BS	
7								
8								
	1	2	3	4	5	6	7	8

White's kashshaptu can move one square in any direction

White enters a start square of 23 and a finish square of 22. The board should now look like this:

1								
2		WK						
3							WG	
4								
5								
6		WS					BS	
7								
8								
	1	2	3	4	5	6	7	8

Figure 6

1								
2	▨	▨ BG	▨					
3	▨	WK	▨				WG	
4	▨	▨	▨					
5								
6		WS					BS	
7								
8								
	1	2	3	4	5	6	7	8

White's kashshaptu can move one square in any direction

White enters a start square of 23 and a finish square of 22. The board should now look like this:

1								
2		WG						
3		WK					WG	
4								
5								
6		WS					BS	
7								
8								
	1	2	3	4	5	6	7	8

White's kashshaptu stays in its original square and the black gisgigir has now become a white gisgigir

Task 1
Adapt the program source code for the `CheckMoveIsLegal` subroutine, so that a kashshaptu (represented by the letter `K`) has the same legal moves as a sarrum.

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

3 3 Your PROGRAM SOURCE CODE for the amended subroutine `CheckMoveIsLegal`.

[1 mark]

Task 2
Adapt the program source code for the `MakeMove` subroutine, so that when a redum is promoted, it becomes a kashshaptu (represented by the symbol `K`) and so that a kashshaptu moves and captures in the way described.

You need to adapt **only** the program source code so that the player with the **White** pieces can promote a redum to a kashshaptu and move a kashshaptu.

Task 3
Test that the changes you have made work:

- run the **Skeleton Program**
- enter `Y` when asked if you want to play the sample game
- enter a start square of `12` and a finish square of `11`
- enter a start square of `41` and a finish square of `31`
- enter a start square of `11` and a finish square of `21`
- enter a start square of `31` and a finish square of `22`
- enter a start square of `11` and a finish square of `22`.

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

3 4 Your amended PROGRAM SOURCE CODE for the subroutine `MakeMove`.

[5 marks]

3 5 SCREEN CAPTURE(S) showing the requested test.

[3 marks]

Turn over ▶

Question 9 This question will further extend the functionality of the **Skeleton Program**.

You can attempt this question regardless of whether or not you produced a solution to **Question 8** that correctly added the kashshaptu piece to the game.

Forsyth-Edwards Notation (FEN) is a standard notation used to describe a particular board position of a chess game. The purpose of FEN is to provide all the information necessary to represent the current state of a game of chess, so that it can be restarted from a particular position.

FEN can be adapted to represent game positions from chess variants like CAPTURE THE SARRUM.

FEN defines a particular game position in one line of text using ASCII characters. This line of text can then be saved into a file or copied for use in another program.
To represent a game position in CAPTURE THE SARRUM, the FEN needs to represent two items of information about the game position – the board state (what pieces are on what squares) and which player's turn it is.

The first item in a FEN record for CAPTURE THE SARRUM is the board state. The board state is represented rank by rank – starting with rank 1 and ending with rank 8. Within each rank the contents of each square are described – starting with file 1 and ending with file 8.

Each piece is identified by a single letter (`S` = sarrum, `M` = marzaz pani, `G` = gisgigir, `E` = etlu, `N` = nabu, `R` = redum, `K` = kashshaptu). White pieces are designated using uppercase letters (`SMGENRK`) and black pieces are designated using lowercase letters (`smgenrk`). Empty squares are represented using the digits 1 to 8 where the digit represents the number of consecutive empty squares.

A `/` character is used to indicate the end of a rank.

After the board state, there will be either a `W` or `B` character to indicate if it is White's or Black's turn.

Figure 7 shows an example board position from a game and the equivalent FEN record.

Figure 8 shows the initial board position and the equivalent FEN record.

Figure 7

1	BG	BE	BN	BM	BS	BN	BE	BG
2	BR	BR	BR		BR		BR	BR
3				BR				
4					WR	BR		
5								
6				WR				
7	WR	WR	WR			WR	WR	WR
8	WG	WE	WN	WM	WS	WN	WE	WG
	1	2	3	4	5	6	7	8

It is Black's turn so the FEN record would be:

```
genmsneg/rrr1r1rr/3r4/4Rr2/8/3R4/RRR2RRR/GENMSNEG/B
```

Turn over for Figure 8

Turn over ▶

Figure 8

1	BG	BE	BN	BM	BS	BN	BE	BG
2	BR	BR	BR	BR	BR	BR	BR	BR
3								
4								
5								
6								
7	WR	WR	WR	WR	WR	WR	WR	WR
8	WG	WE	WN	WM	WS	WN	WE	WG
	1	2	3	4	5	6	7	8

It is White's turn so the FEN record would be:

```
genmsneg/rrrrrrrr/8/8/8/8/RRRRRRRR/GENMSNEG/W
```

Task 1
Create a new subroutine, `GenerateFEN`, which takes two parameters (the board and whose turn it is) and creates a string containing the FEN record for the current state of a CAPTURE THE SARRUM game. It should return this string to the calling routine.
You may choose whether to make the new subroutine a function or a procedure.

You are likely to get some marks for this task, even if your subroutine is only partially working.

It does not matter if your new subroutine will work correctly for positions containing a kashshaptu (from **Question 8**).

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

3 6 Your PROGRAM SOURCE CODE for the new subroutine `GenerateFEN`.
[13 marks]

Task 2
Adapt the MAIN PROGRAM BLOCK so that the FEN record returned by the `GenerateFEN` subroutine is displayed **before** asking the player to enter their move.

Task 3
Test your program works by conducting the following test:

- run the **Skeleton Program**
- enter `Y` when asked if you want to play the sample game.

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

3 7 Your PROGRAM SOURCE CODE for the amended MAIN PROGRAM BLOCK.
[3 marks]

3 8 SCREEN CAPTURE(S) showing the requested test.
[2 marks]

END OF QUESTIONS

There are no questions printed on this page

There are no questions printed on this page

There are no questions printed on this page