General Certificate of Education
Advanced Subsidiary Examination
June 2012

Computing COMP1

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

Friday 25 May 2012 9.00 am to 11.00 am

You will need to:
- access the Electronic Answer Document
- refer to the Preliminary Material and the 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
- Before the start of the examination make sure your **Centre Number, Candidate Name** and **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 – 20 minutes
  Section C – 15 minutes
  Section D – 50 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 the Electronic Answer Document.

SECTION A

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

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

Question 1

Table 1 is a partially complete representation of the rules for adding together two bit values. The first two columns represent the two bit values to add. The first row has been completed and represents the binary addition rule `0 + 0 = 0`. Carry occurs when the answer cannot be stored in 1 bit.

Table 1

		Answer	Carry
0	0	0	0
0	1		
1	0		
1	1		

0 1 Complete **Table 1** to show the **Answer** and **Carry** values for the given binary addition rules.

*Copy the cells in **Table 1** that contain your answer into the Electronic Answer Document.* *(3 marks)*

Question 2

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.

0 2 Using 7 bits, express the ASCII code for the character '2' in binary. *(1 mark)*

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

0 3 Using odd parity, what 8-bit code is sent for the numeric character '0'? *(2 marks)*

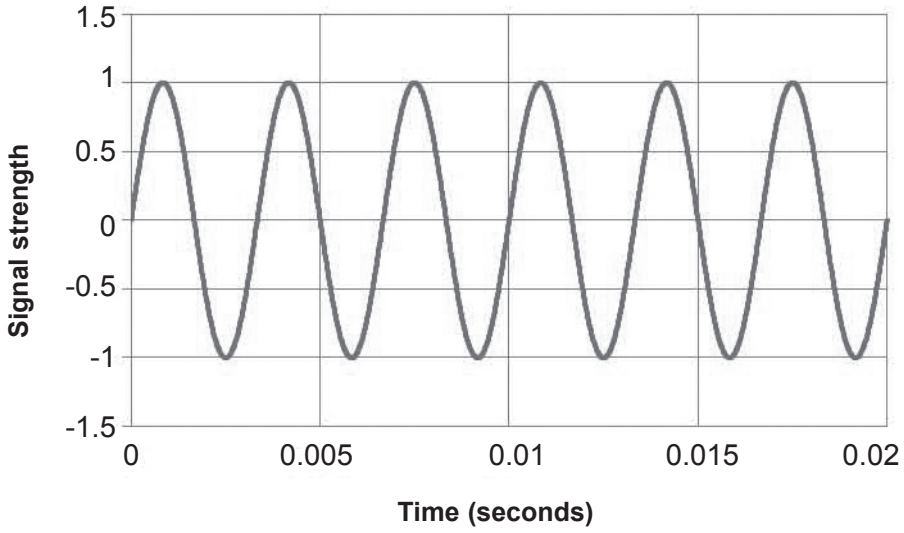
Hamming code is an alternative to the use of a single parity bit.

0 4 State **one** advantage of using Hamming code instead of a single parity bit. *(1 mark)*

Question 3

To record sound a computer needs to convert the analogue sound signal into a digital form. During this process samples of the analogue signal are taken. **Figure 1** shows part (0.02 seconds) of an analogue sound wave.

Figure 1



The **frequency** of an analogue sound wave is determined by how many waves of oscillation occur per second and is measured in Hertz (Hz) – the number of waves of oscillation per second.

0 5 If the part of the analogue sound shown in **Figure 1** is the highest frequency in the entire sound to be sampled, what is the **minimum sampling rate** (in Hz) that should be used?

Use the space below for rough working - then copy the answer, and your working out, to your Electronic Answer Document. You may get some marks for your working even if your answer is incorrect if you include the working in your Electronic Answer Document.

(2 marks)

0 6 Describe clearly the steps taken by an ADC (analogue-to-digital converter) in the conversion of an analogue sound wave to an equivalent digital signal. *(3 marks)*

MIDI is an alternative method for storing sound digitally that does not use sound waves; instead, information about each musical note is stored.

0 7 State **one** advantage of using the MIDI representation for storing sound digitally.

(1 mark)

Turn over ▶

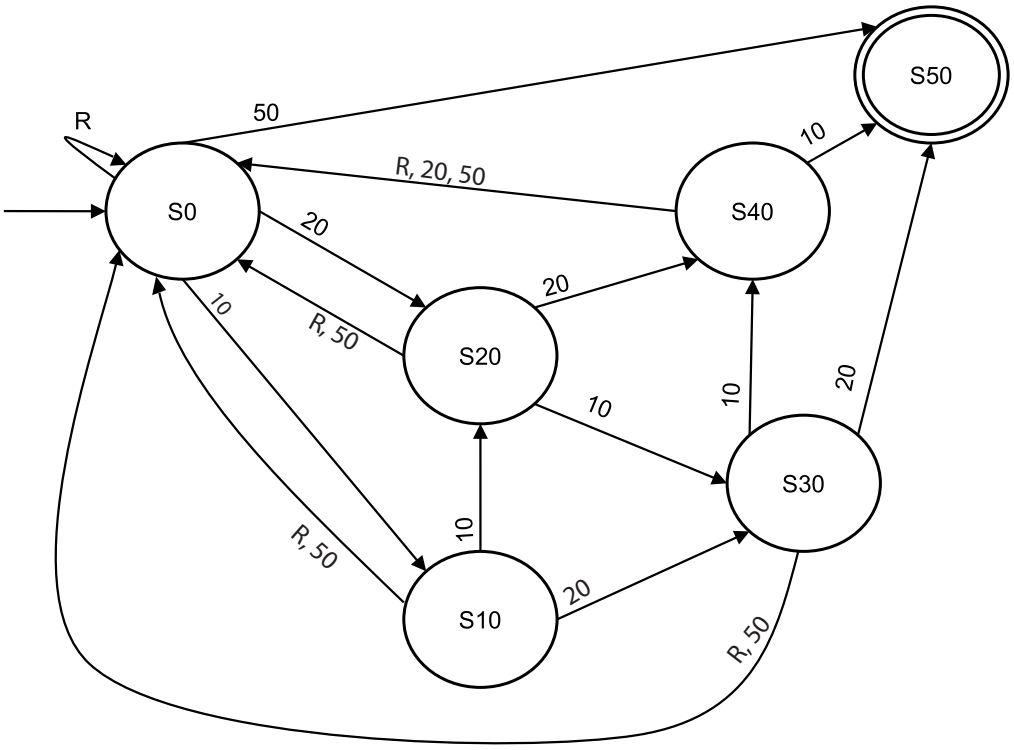
0 8 State an item of data, other than the note itself, that might be stored about a musical note in a MIDI file. *(1 mark)*

Question 4

Figure 2 shows the state transition diagram of a finite state machine (FSM) used to control a vending machine.

The vending machine dispenses a drink when a customer has inserted exactly 50 pence. A transaction is cancelled and coins returned to the customer if more than 50 pence is inserted or the reject button (R) is pressed. The vending machine accepts 10, 20 and 50 pence coins. Only one type of drink is available.
The only acceptable inputs for the FSM are 10, 20, 50 and R.

Figure 2



0 9 An FSM can be represented as a state transition diagram or as a state transition table. **Table 2** is an incomplete state transition table for part of **Figure 2**.

Complete the missing sections of the four rows of **Table 2**.

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

Table 2

Original state	Input	New state
S0	10	S10
S0		
S0		
S0		

(3 marks)

There are different ways that a customer can provide **exactly three** inputs that will result in the vending machine dispensing a drink. Three possible permutations are "20, 10, 20", "10, R, 50" and "10, 50, 50".

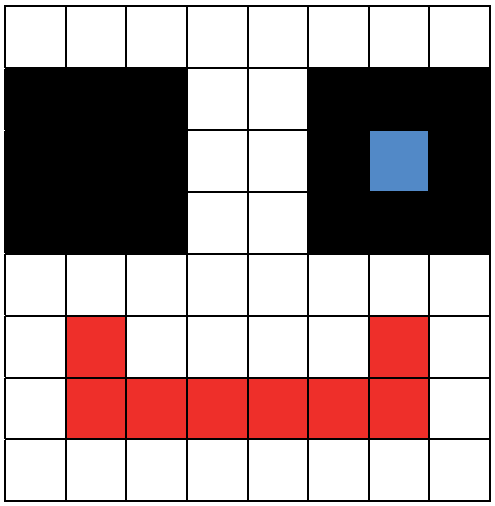
1 0 List **four** other possible permutations of **exactly three** inputs that will be accepted by the FSM shown in **Figure 2**. *(4 marks)*

Turn over for the next question

Question 5

A bitmapped image consists of pixels. **Figure 3** shows a bitmapped representation of an image of a winking, happy face consisting of red, blue, black and white pixels only.

Figure 3



1 1 Why must at least two bits be used to represent each pixel? *(1 mark)*

The second line of pixels (from the top) shown in **Figure 3** has been represented in a computer's memory as the bit pattern `1111 1100 0011 1111`. A black pixel is coded as `11`.

1 2 Suggest a suitable 16-bit bit pattern that could be used to represent the third line of pixels (from the top) in **Figure 3**.

Type your answer into the table provided in the Electronic Answer Document. *(2 marks)*

1 3 What, in bytes, is the minimum file size for the bitmapped image in **Figure 3**?

Use the space below for rough working - then copy the answer, and your working out, to your Electronic Answer Document. You may get some marks for your working even if your answer is incorrect if you include the working in your Electronic Answer Document.

(3 marks)

Instead of representing the face as a bitmapped image, vector graphics could have been used.

1 4 State **three** items of data that would need to be stored about an eye object, similar to those shown in the image in **Figure 3**, if it is to be represented using vector graphics. *(3 marks)*

1 5 Describe **two** advantages of using vector graphics instead of bitmaps to represent an image. *(2 marks)*

Turn over for the next section

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**.

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

Question 6

Create a folder/directory **Question6** for your new program.

The algorithm, represented as a flowchart in **Figure 4**, and the variable table, **Table 3**, describe the converting of a 4-bit binary value into denary.

Figure 4

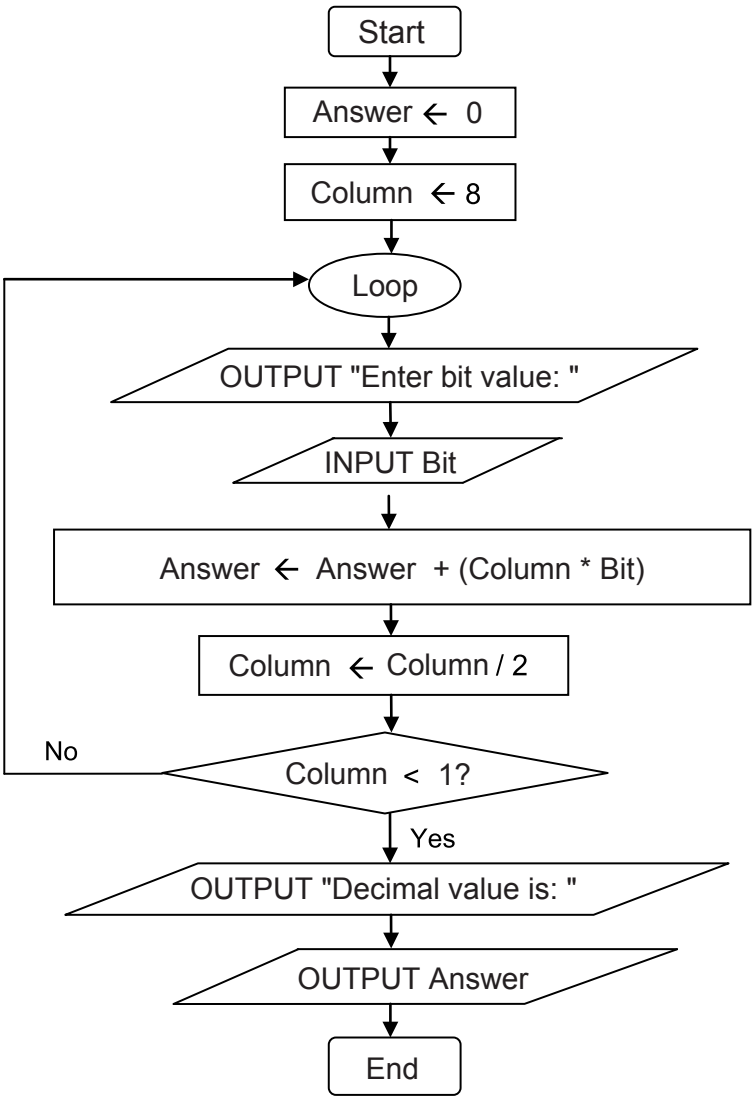


Table 3

Identifier	Data type	Purpose
`Column`	`Integer`	Stores the place value (column heading)
`Answer`	`Integer`	Stores the denary value equivalent to the bit pattern entered by the user
`Bit`	`Integer`	Stores a `0` or `1` entered by the user

What you need to do

Write a program for the above algorithm.

Test the program by showing the result of entering the values `1`, `1`, `0`, `1` (in that order).

Save the program in your new **Question6** folder/directory.

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

16 Your PROGRAM SOURCE CODE. *(11 marks)*

17 SCREEN CAPTURE(S) for the test described above. *(3 marks)*

18 What is the largest denary number that could be output by the algorithm represented by the flowchart in **Figure 4**? *(1 mark)*

The algorithm represented by the flowchart in **Figure 4** can convert sixteen different bit patterns into denary.

19 If the symbol Column ← 8 is changed to Column ← 16 how many **more** bit patterns could be converted into denary? *(1 mark)*

When developing a new system the stages of the systems development life cycle could be followed.

20 At which stage of the systems development life cycle would the flowchart in **Figure 4** have been created? *(1 mark)*

21 At which stage of the systems development life cycle would the algorithm represented by the flowchart in **Figure 4** be automated using a programming language? *(1 mark)*

SECTION C

You are advised to spend no more than **15 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 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 7

State the name of an identifier for:

2 2 a user-defined subroutine that has only one parameter. *(1 mark)*

2 3 a user-defined subroutine whose only action is to produce output to the screen. *(1 mark)*

2 4 a variable that has a stepper role. *(1 mark)*

2 5 an array variable. *(1 mark)*

Look at the repetition structure in the `SetPositionOfItem` subroutine.

2 6 Describe the circumstances under which this structure in the **Skeleton Program** will stop repeating. *(3 marks)*

Look at the `SetUpGame` subroutine.

2 7 Why has a `For` loop been chosen for the repetition structure? *(1 mark)*

The `For` loop repeats `NoOfTrap` times.

2 8 Why has a named constant been used instead of the numeric value 2? *(1 mark)*

When a game is saved it is stored as a binary file. A text file could have been used instead.

2 9 Describe a difference between the way that data are stored in a binary file and the way that data are stored in a text file. *(2 marks)*

The subroutines in the **Skeleton Program** avoid the use of global variables – they use local variables and parameter passing instead.

3 0 State **two** reasons why subroutines should, ideally, not use global variables. *(2 marks)*

Figure 5 shows a pseudo-code representation of the part of the `PlayGame` subroutine that is used to check if the player has triggered one of the traps in the cavern.

Figure 5

```
MonsterAwake ← CheckIfSameCell(PlayerPosition,
               TrapPositions[1])
If Not MonsterAwake
   Then MonsterAwake ← CheckIfSameCell(PlayerPosition,
                       TrapPositions[2])
EndIf
```

3 1 Why is it necessary that the check for the triggering of the second trap is inside the selection structure? *(2 marks)*

Turn over for the next section

SECTION D

You are advised to spend no more than **50 minutes** on this section.

Enter 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.

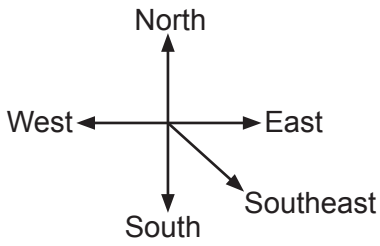
Question 8

This question refers to the subroutines `DisplayMoveOptions`, `CheckValidMove` and `MakeMove`.

The player can currently move in four directions – north, south, west and east. The player is to be allowed to move diagonally.

Adapt the program source code for the subroutines `DisplayMoveOptions`, `CheckValidMove` and `MakeMove` so that there is a fifth direction – southeast (as shown in **Figure 6**) – that can be selected by entering a "`D`".

Figure 6



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

3 2	Your amended PROGRAM SOURCE CODE for the subroutine `DisplayMoveOptions`. *(1 mark)*
3 3	Your amended PROGRAM SOURCE CODE for the subroutine `MakeMove`. *(3 marks)*
3 4	Your amended PROGRAM SOURCE CODE for the subroutine `CheckValidMove`. *(1 mark)*
3 5	SCREEN CAPTURE(S) for a test run showing the correct working of the new move option being selected and the player moving to the southeast. *(2 marks)*

Question 9

This question refers to the subroutines `CheckValidMove` and `PlayGame`.

The **Skeleton Program** currently does not make all the checks needed to ensure that the move entered by a player is an allowed move. It should **not** be possible to make a move that takes a player outside of the 7×5 cavern grid.

The **Skeleton Program** needs to be adapted so that it prevents a player from moving north if they are at the northernmost end of the cavern.

The subroutine `CheckValidMove` needs to be adapted so that it returns a value of `False` if a player attempts to move north when they are at the northernmost end of the cavern.

The subroutine `PlayGame` needs to be adapted so that it displays an error message to the user if an illegal move is entered. The message should state "`That is not a valid move, please try again`".

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

3 6	Your amended PROGRAM SOURCE CODE for the subroutine `PlayGame`.	*(3 marks)*
3 7	Your amended PROGRAM SOURCE CODE for the subroutine `CheckValidMove`.	*(4 marks)*
3 8	SCREEN CAPTURE(S) for a test run showing a player trying to move north when they are at the northernmost end of the cavern.	*(1 mark)*

Turn over for the next question

Question 10

This question refers to the `PlayGame` subroutine and will extend the functionality of the game.

The number of moves made by a player in a game of MONSTER! will be tracked. A variable called `NoOfMoves` will be used to store the number of moves made by a player.

The final number of moves made will be displayed to the user at the end of the game. At the end of the game, either the player will have found the flask or the player will have been eaten by the monster.

If they have found the flask then a message should be displayed saying `"The number of moves you took to find the flask was X"` – where `X` is the value of `NoOfMoves`.

If they were eaten then a message should be displayed saying `"The number of moves that you survived in the cavern for was X"` – where `X` is the value of `NoOfMoves`.

Task 1
Create a new variable, of an appropriate data type, called `NoOfMoves`. **At the start of a game** an initial value of `0` should be assigned to the `NoOfMoves` variable.

Task 2
The value of `NoOfMoves` needs to be incremented **after** a player has completed a move in the cavern.

Task 3
Adapt the relevant subroutine(s) so that the correct messages are displayed at the end of a game of MONSTER!

Task 4 – Test 1
Test that the changes you have made work by conducting the following test:

- Play the training game
- Move south
- Move south
- Move east

Task 5 – Test 2
Test that the changes you have made work by conducting the following test:

- Play the training game
- Move south
- Move west

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

3 9	Your amended PROGRAM SOURCE CODE for the `PlayGame` subroutine and (if relevant) the PROGRAM SOURCE CODE for any other subroutine(s) you have amended. *(5 marks)*
4 0	SCREEN CAPTURE(S) showing the result of **Test 1**. *(1 mark)*
4 1	SCREEN CAPTURE(S) showing the result of **Test 2**. *(1 mark)*

Turn over for the next question

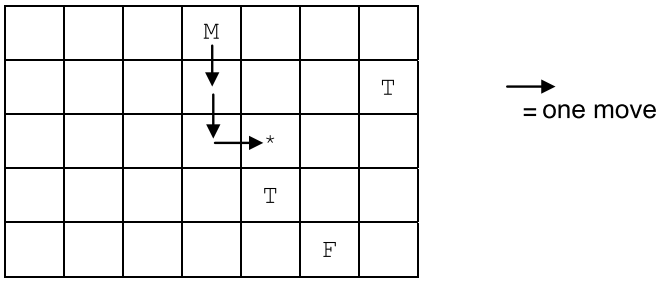
Question 11

This question will extend the functionality of the game.

The noise made by the sleeping monster does not help the player work out in which direction the monster is – however, it gets louder as the player moves nearer to the monster and gets quieter as the player moves further away from the monster.

The game is to be adapted so that after the move options have been displayed (but before the user enters their move) a message is displayed stating the distance between the monster and the player.

The *distance* between the monster and the player is measured by the number of cells the monster would have to move into in order to get to the cell currently occupied by the player. For example, at the start of the training game (**Figure 4** in the **Preliminary Material**, reproduced below) the distance would be 3 as the monster would have to move into 3 cells in order to get the player's cell.



Additional marks will be awarded in Question 11 for writing code that demonstrates good practice by ensuring subroutines are self-contained and make use of interfaces.

Task 1

Create a new subroutine, `CalculateDistance`, which works out the distance between the cell currently occupied by the monster and the cell currently occupied by the player. It should then return this calculated value to the calling routine.

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

4 2 Your PROGRAM SOURCE CODE for the new subroutine `CalculateDistance`. *(7 marks)*

Task 2

Adapt the `PlayGame` subroutine so that it displays (after the move options have been shown) the message "`Distance between monster and player: X`" – where *X* is the distance between the monster and the player.

Test that your program works by loading the training game and showing that:

- the correct distance is displayed before the player's first move
- the correct distance is displayed after the player's first move, one cell to the north
- the correct distance is displayed after the player's third move, both second and third moves are westwards.

Evidence that you need to provide

Include the following in your Electronic Answer Document.

4 3	Your amended PROGRAM SOURCE CODE for the `PlayGame` subroutine. *(3 marks)*
4 4	SCREEN CAPTURE(S) showing the distance message and the cavern at the start of the training game, before the player's first move. *(1 mark)*
4 5	SCREEN CAPTURE(S) showing the distance message and the cavern after the player has moved one cell to the north. *(1 mark)*
4 6	SCREEN CAPTURE(S) showing the distance message and the cavern after the player has then moved two cells to the west. *(1 mark)*

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