General Certificate of Education
Advanced Subsidiary Examination
June 2013

Computing COMP1

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

Monday 3 June 2013 1.30 pm to 3.30 pm

You will need to:
- access the Electronic Answer Document
- refer to the Preliminary Material, Data File 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 and Data File
  - 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).

A

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 – 25 minutes
  Section C – 10 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.

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 What is the denary equivalent of the hexadecimal number A7?

You may 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 2 Represent the denary value 7.625 as an **unsigned binary fixed point** number, with 4 bits before and 4 bits after the binary point.

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

(2 marks)

0 3 Represent the denary value -18 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 4 What is the **largest positive denary value** that can be represented using **8-bit two's complement binary**?

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

(1 mark)

0 5 Describe how **8-bit two's complement binary** can be used to subtract one number from another number. In your answer you must show how the calculation 23 – 48 would be completed using the method that you have described.

You may use the space below for rough working.

(4 marks)

Question 1 continues on the next page

Turn over ▸

Figure 1 shows a state transition diagram for a finite state machine (FSM).

Table 1 shows the outputs produced by the finite state machine in **Figure 1** for some possible input strings. Some of the outputs are missing from **Table 1**. Input strings are processed starting with the right-most bit.

Figure 1

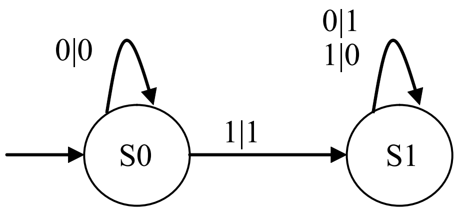


Table 1

Input string	Output string
00010011	11101101
00010010	**(a)**
00010100	11101100
00010101	**(b)**

0 6 What output string should be in position **(a)** in the table? *(1 mark)*

0 7 What output string should be in position **(b)** in the table? *(1 mark)*

0 8 What is the purpose of the finite state machine shown in **Figure 1**? *(1 mark)*

A finite state machine can be represented as a state transition diagram or as a state transition table. **Table 2** is an incomplete state transition table for **Figure 1**.

0 9 *Complete the **unshaded** cells in the table in the Electronic Answer Document that correspond to the unshaded cells in **Table 2** below.*

Table 2

Input	Original state	Output	New state
0	S0	0	S0
1		1	S1
0	S1		S1

(3 marks)

Question 2

One character encoding scheme is Unicode. An alternative character encoding scheme is ASCII.

1 0 State **one** difference between Unicode and ASCII. *(1 mark)*

7-bit ASCII codes are often transmitted using 8 bits, with a single parity bit added in the most significant bit to help with error detection.

1 1 Explain how the even parity system works. Include a description of the roles of the sending device and the receiving device during transmission. *(4 marks)*

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

1 2 State **one** advantage of using Hamming code instead of the simple parity bit system. *(1 mark)*

Figure 2 shows the data bits and some of the parity bits that will be transmitted for the ASCII representation of the numeric character '3'. Even parity Hamming code is used for the transmission. One of the parity bits in the Hamming code has not been calculated.

Figure 2

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

1 3 What value should be given to the parity bit in bit position 2? *(1 mark)*

Four of the bit positions shown in **Figure 2** are being used as parity bits. Bit position 2 is one of these parity bits.

1 4 What are the bit positions of the other **three** parity bits? *(1 mark)*

Turn over for the next question

Question 3

A performance by a music band is to be recorded and distributed on CD.

Figure 3 shows three samples stored in a computer's memory that have been taken from an analogue signal as part of the recording process. A sampling rate of 44,000Hz (Hertz) has been used.

1Hz is one sample per second.

Figure 3

`0000 0001 1000 1110`
`0000 0001 1000 1110`
`0000 0001 1000 0011`

1 5 What sampling resolution has been used? *(1 mark)*

1 6 If the original analogue signal lasts 100 seconds, how many bytes of storage will be required to store all the samples taken in the recording process? *(3 marks)*

You may 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.

The average human can hear frequencies up to 20,000Hz (Hertz).

1 7 Explain why a sampling rate of 44,000Hz has been chosen for the recording. *(2 marks)*

The CD recording is processed to create a version of the performance that can be downloaded from the band's website.

The sound quality of the version of the recording stored on the web server is not as good as the sound quality of the CD version.

1 8 State **one** possible cause of this reduction in sound quality. *(1 mark)*

There are no questions printed on this page

Section B begins on the next page

Turn over ▶

Section B

You are advised to spend no more than **25 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 simple two player game. Player One chooses a whole number between 1 and 10 (inclusive) and then Player Two tries to guess the number chosen by Player One. Player Two gets up to five attempts to guess the number. Player Two wins the game if they correctly guess the number, otherwise Player One wins the game.

Note that in **Figure 4**, the symbol <> means "is not equal to".

Figure 4

```
OUTPUT "Player One enter your chosen number: "
INPUT NumberToGuess
WHILE NumberToGuess < 1 OR NumberToGuess > 10 DO
  OUTPUT "Not a valid choice, please enter another number: "
  INPUT NumberToGuess
ENDWHILE
Guess ← 0
NumberOfGuesses ← 0
WHILE Guess <> NumberToGuess AND NumberOfGuesses < 5 DO
  OUTPUT "Player Two have a guess: "
  INPUT Guess
  NumberOfGuesses ← NumberOfGuesses + 1
ENDWHILE
IF Guess = NumberToGuess
  THEN OUTPUT "Player Two wins"
  ELSE OUTPUT "Player One wins"
```

Table 3

Identifier	Data type	Purpose
`NumberToGuess`	`Integer`	Stores the number entered by Player One
`NumberOfGuesses`	`Integer`	Stores the number of guesses that Player Two has made so far
`Guess`	`Integer`	Stores the most recent guess made by Player Two

What you need to do

Write a program for the above algorithm.

Test the program by conducting the tests **Test 1** and **Test 2**.

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

Test 1
Test that your program works correctly by conducting the following test:

- Player One enters the number `0`
- Player One enters the number `11`
- Player One enters the number `5`
- Player Two enters a guess of `5`

Test 2
Test that your program works correctly by conducting the following test:

- Player One enters the number `6`
- Player Two enters guesses of `1`, `3`, `5`, `7`, `10`

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

1 9 Your PROGRAM SOURCE CODE. *(13 marks)*

2 0 SCREEN CAPTURE(S) showing the result of **Test 1**. *(4 marks)*

2 1 SCREEN CAPTURE(S) showing the result of **Test 2**. *(3 marks)*

Turn over ▸

Part of the algorithm from **Figure 4** is shown in **Figure 5**.
Note that in **Figure 5**, the symbol <> means "is not equal to".

Figure 5

```
WHILE Guess <> NumberToGuess AND NumberOfGuesses < 5 DO
  OUTPUT "Player Two have a guess: "
  INPUT Guess
  NumberOfGuesses ← NumberOfGuesses + 1
ENDWHILE
```

2 2 Explain why a `WHILE` repetition structure was chosen instead of a `FOR` repetition structure for the part of the algorithm shown in **Figure 5**. *(1 mark)*

There are no questions printed on this page

Section C begins on the next page

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

Throughout Question 5, you must be careful to copy and paste or type accurately the names of identifiers from the **Skeleton Program**.

State the name of an identifier for:

2 3 a variable used to store whole numbers. *(1 mark)*

2 4 a user-defined subroutine that has exactly **three** parameters. *(1 mark)*

2 5 a built-in function with exactly **one** parameter that returns an integer value. *(1 mark)*

2 6 Give an example of an assignment statement from the **Skeleton Program** where a variable is assigned an empty string. *(1 mark)*

Look at the *option j* in the main program block.

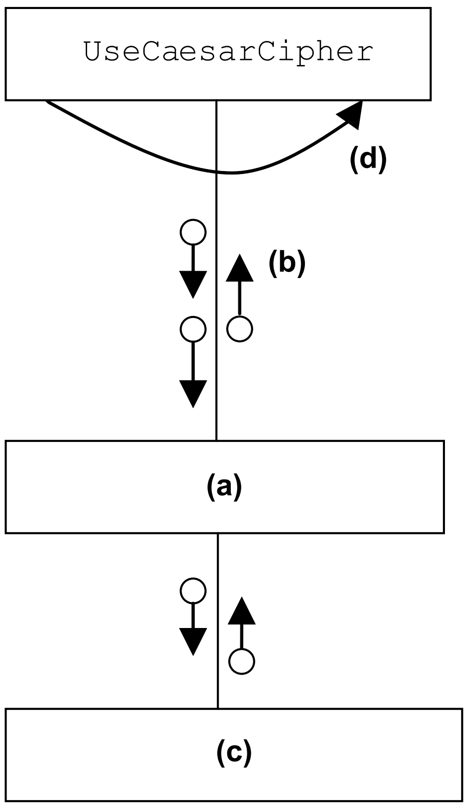
2 7 Explain why `AmountToShift` needs to be assigned the value of `-GetKeyForCaesarCipher`. *(2 marks)*

Look at the `ApplyShiftToASCIICodeForCharacter` subroutine.

2 8 Explain, using an example, why `Mod 26` is used when calculating `NewASCIICode`. *(2 marks)*

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

Figure 6



2 9 What should be written in box **(a)** in **Figure 6**? *(1 mark)*

3 0 How should the arrow **(b)** in **Figure 6** be labelled? *(1 mark)*

3 1 What should be written in box **(c)** in **Figure 6**? *(1 mark)*

3 2 How should the curved arrow **(d)** in **Figure 6** be labelled? *(1 mark)*

Section D begins on the next page

Turn over ▸

Section D

You are advised to spend no more than **50 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.

You are advised to save your program at regular intervals.

Question 6

This question refers to the subroutine `GetKeyForCaesarCipher`.

If the user chooses to encrypt/decrypt using a Caesar cipher they are prompted to enter a value to indicate the number of characters to shift the plaintext alphabet. This value is then stored in the variable `Key`.

Add a validation check to the subroutine `GetKeyForCaesarCipher` so that it checks that the value entered by the user is between 1 and 25 inclusive. If the value entered is **not** in this range then `Key` should be given a default value of 1 and the message "`Invalid key – a default value has been used instead`" displayed.

Test that the changes you have made work:

- run the **Skeleton Program**
- select *option a* from the menu
- enter `Zebra` as the plaintext
- select *option g* from the menu and enter `0` as the amount to shift the plaintext
- select *option g* from the menu and enter `27` as the amount to shift the plaintext
- select *option g* from the menu and enter `4` as the amount to shift the plaintext.

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

33 Your amended PROGRAM SOURCE CODE for the subroutine `GetKeyForCaesarCipher`. *(4 marks)*

34 SCREEN CAPTURE(S) showing the results of testing the subroutine with values of `0`, `27` and `4`. *(3 marks)*

Question 7

If an encrypted message has been intercepted by someone who does not know the key to decrypt the message then they might try to crack the encryption. One way of doing this is to try lots of different decryption methods and see if any of them produce a sensible-looking result.

This question will extend the functionality of the **Skeleton Program** so that there is a new option that attempts to crack an encrypted message by trying to decrypt it using all rail fence sizes greater than 1 and less than the length of the message.

Figure 7 shows an example of cracking a rail fence encrypted message using this method.

Figure 7

If the ciphertext is `sceert` and the message had been encrypted using a rail fence cipher then the decrypted message would be one of these possible plaintexts:

`secret` (rail fence size of 2)
`sercet` (rail fence size of 3)
`sertce` (rail fence size of 4)
`seertc` (rail fence size of 5).

A rail fence size of either 1 or 6 (the length of the encrypted message) would result in the plaintext being the same as the original ciphertext.

Looking at these possibilities the user would then be able to see that the original plaintext message was `secret`.

Question 7 continues on the next page

Turn over ▶

A new option (*option m*) is to be added to the menu. This option will use each rail fence size in turn on the ciphertext and display the resulting plaintext to the user. The user will then be able to look at all the possible plaintexts and see if any result in a sensible message.

Task 1
Change the `DisplayMenu` subroutine so that it displays the new option
`"  m.  Brute force rail fence solver"`

Task 2
Adapt the main program block so that the case where the user selects *option m* is dealt with. When the user selects this option there should be a call to the `DecryptUsingRailFence` subroutine (with `Ciphertext` as a parameter) followed by a call to the `DisplayPlaintext` subroutine (with the value returned by `DecryptUsingRailFence` as a parameter). To obtain full marks for **Question 7** these two subroutine calls will need to be within a repetition structure that will result in all possible plaintext messages being displayed.

You might find the example described in **Figure 7** (using ciphertext of `sceert`) helpful when testing that the modifications you have made to the **Skeleton Program** are working correctly.

Task 3
Test that the changes you have made work by conducting the following test:

- run the **Skeleton Program**
- select *option d* from the menu
- enter `RLNAFCIEE` as the ciphertext
- select *option m* from the menu.

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

3 5 Your amended PROGRAM SOURCE CODE for the subroutine `DisplayMenu`. *(1 mark)*

3 6 Your amended PROGRAM SOURCE CODE for the main program block. *(5 marks)*

3 7 SCREEN CAPTURE(S) showing the results of testing the new menu option. Test by entering `RLNAFCIEE` as the value for the ciphertext and then selecting *option m* on the menu. *(3 marks)*

Question 8

This question will extend the functionality of the **Skeleton Program**.

In the **Skeleton Program** there is currently only one steganography option – where the characters are hidden in the text at regular intervals (every *n*th character). There are other ways of concealing a message in a piece of text – one possibility is to place the characters of the message at intervals determined by a more complex sequence of numbers than every *n*th character.

The Fibonacci numbers are the numbers in the following sequence:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

- The 1st Fibonacci number is 1.
- The 2nd Fibonacci number is 1.
- Each subsequent Fibonacci number is the sum of the two previous Fibonacci numbers. For example, the 3rd Fibonacci number is the sum of the 1st and 2nd Fibonacci numbers (1 + 1 = 2), the 6th Fibonacci number is sum of the 4th and 5th Fibonacci numbers (3 + 5 = 8).

The *n*th Fibonacci number is equal to the sum of the (*n*-1)th and (*n*-2)th Fibonacci numbers. **Table 4** contains a summary of the Fibonacci number sequence.

Table 4

Sequence number (*n*)	**Fibonacci number**
1	1
2	1
3	2
4	3
5	5
6	8
7	13
...	...
n	(*n*-1)th Fibonacci number + (*n*-2)th Fibonacci number

The **Skeleton Program** is to be extended so that it can be used to find messages that have been hidden in a text file at positions determined by the Fibonacci number sequence.

Turn over ▶

Task 1
Change the `DisplayMenu` subroutine so that it displays the new option
`"   o. Fibonacci sequence (text from file)"`

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

3 8 Your amended PROGRAM SOURCE CODE for the subroutine `DisplayMenu`.
(1 mark)

Task 2
Create a new subroutine, `GetNthFibonacciNumber`, which takes an integer parameter `n` and calculates the *n*th Fibonacci number. For example:

`GetNthFibonacciNumber(1)` should return 1
`GetNthFibonacciNumber(2)` should return 1
`GetNthFibonacciNumber(3)` should return 2
`GetNthFibonacciNumber(6)` should return 8

You will get some marks for this task even if your subroutine only works for some values of *n*.

You are **not** required, but might find it useful, to add an extra menu option that calls the `GetNthFibonacciNumber` subroutine to test that it works as expected.

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

3 9 Your PROGRAM SOURCE CODE for the new subroutine `GetNthFibonacciNumber`.
(10 marks)

The algorithm, represented using pseudo-code in **Figure 8**, describes the functionality of what should happen when the user selects *option o*.

Figure 8

```
Plaintext ← ""
N ← 2
Input start and end positions to use with file diary.txt
WHILE StartPosition <= EndPosition DO
  Plaintext ← Plaintext + GetTextFromFile(StartPosition,
                StartPosition)
  StartPosition ← StartPosition + GetNthFibonacciNumber(N)
  N ← N + 1
ENDWHILE
Display the Plaintext
```

Task 3
Adapt the MAIN PROGRAM BLOCK so that the case where the user selects *option o* is dealt with.

Task 4
There is a message in the **Data File diary.txt** that has been hidden using the first seven Fibonacci numbers.

Test that your program works, and that it finds the correct hidden message, by:

- selecting *option o* from the menu
- entering a start position of `665`
- entering an end position of `697`

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

4 0 Your amended PROGRAM SOURCE CODE for the main program block. *(6 marks)*

4 1 SCREEN CAPTURE(S) for a test run showing:

- *option o* being selected by the user
- a start position of `665` entered by the user
- an end position of `697` entered by the user
- the hidden message displayed to the user. *(2 marks)*

END OF QUESTIONS

There are no questions printed on this page