EXETER MATHEMATICS SCHOOL

SERVING CORNWALL, DEVON, DORSET AND SOMERSET

Computer Science End of Year 12 Assessment

Paper 1

Time Allowed
2 hours and 15 minutes

Materials
- For this paper you must have 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**.

Instructions
- Type the information required on the front of your **Electronic Answer Document**.
- Enter your answers into the **Electronic Answer Document**.
- Answer **all** questions.
- 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).
- Tie together all your printed **Electronic Answer Document** pages and hand them to the invigilator.

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 **not** allowed to use a calculator.

Total Marks 87

Advice
- You are advised to spend time on each section as follows:
  - Section A – 20 minutes
  - Section B – 20 minutes
  - Section C – 25 minutes
  - Section D – 60 minutes
- Save your work at regular intervals.

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.

---

1 Regular expressions can be used to search for strings.

For each of the following regular expressions, describe the set of strings that they would match.

1.1

`b*c` **[1 mark]**

1.2

`b?c` **[1 mark]**

1.3

`(b|c)+d` **[1 mark]**

1.4

Write a regular expression that matches the letter `b`, followed by zero or more occurrences of the string `cd` followed by either a single letter `e` or the string `fg`. **[2 marks]**

2 A list data structure can be represented using an array.

The pseudocode algorithm in **Figure 7** can be used to carry out one useful operation on a list.

Figure 7

```
p ← 1
If ListLength > 0 Then
  While p <= ListLength And List[p] < New Do
    p ← p + 1
  EndWhile
  For q ← ListLength DownTo p Do
    List[q + 1] := List[q]
  EndFor
EndIf
List[p] ←  New
ListLength ← ListLength + 1
```

2.1 The initial values of the variables for one particular execution of the algorithm are shown in the trace table below, labelled **Table 3**.

Complete the trace table for the execution of the algorithm.

Table 3

				List				
ListLength	New	p	q	[1]	[2]	[3]	[4]	[5]
4	38	-	-	9	21	49	107	

(4 marks)

2.2

Describe the purpose of the algorithm in **Figure 7**.

....

....

(1 mark)

3 A Turing machine has been designed to recognise palindromic binary numbers, ie numbers such as `101` and `0110` that read the same from left to right as from right to left.

The machine has states S_B, S_0, S_1, S_{C0}, S_{C1}, S_L, S_Y and S_N.
S_B is the start state and S_Y and S_N are the stop states.

The machine stores data on a single tape which is infinitely long in one direction. The machine's alphabet is 0, 1 and □, where □ is the symbol used to indicate a blank cell on the tape. The machine will enter state S_Y if the value represented on the tape is a palindromic binary number, otherwise it will enter state S_N.

The transition rules for this Turing machine can be expressed as a transition function δ. Rules are written in the form:

$$\delta(\text{Current State, Input Symbol}) = (\text{Next State, Output Symbol, Movement})$$

So, for example, the rule:

$$\delta(S_B, 0) = (S_0, \square, \rightarrow)$$

means:

IF the machine is currently in state S_B AND the input symbol read from the tape is 0

THEN the machine should change to state S_0, write a blank symbol (□) to the tape and move the read/write head one cell to the right

The machine's transition function, δ, is defined by:

$$\begin{array}{lll@{\qquad\qquad}lll}
\delta(S_B, 0) & = & (S_0, \square, \rightarrow) & \delta(S_{C0}, 0) & = & (S_L, \square, \leftarrow) \\
\delta(S_B, 1) & = & (S_1, \square, \rightarrow) & \delta(S_{C0}, 1) & = & (S_N, 1, \leftarrow) \\
\delta(S_B, \square) & = & (S_Y, \square, \rightarrow) & \delta(S_{C0}, \square) & = & (S_Y, \square, \rightarrow) \\
 & & & & & \\
\delta(S_0, 0) & = & (S_0, 0, \rightarrow) & \delta(S_{C1}, 0) & = & (S_N, 0, \leftarrow) \\
\delta(S_0, 1) & = & (S_0, 1, \rightarrow) & \delta(S_{C1}, 1) & = & (S_L, \square, \leftarrow) \\
\delta(S_0, \square) & = & (S_{C0}, \square, \leftarrow) & \delta(S_{C1}, \square) & = & (S_Y, \square, \rightarrow) \\
 & & & & & \\
\delta(S_1, 0) & = & (S_1, 0, \rightarrow) & \delta(S_L, 0) & = & (S_L, 0, \leftarrow) \\
\delta(S_1, 1) & = & (S_1, 1, \rightarrow) & \delta(S_L, 1) & = & (S_L, 1, \leftarrow) \\
\delta(S_1, \square) & = & (S_{C1}, \square, \leftarrow) & \delta(S_L, \square) & = & (S_B, \square, \rightarrow)
\end{array}$$

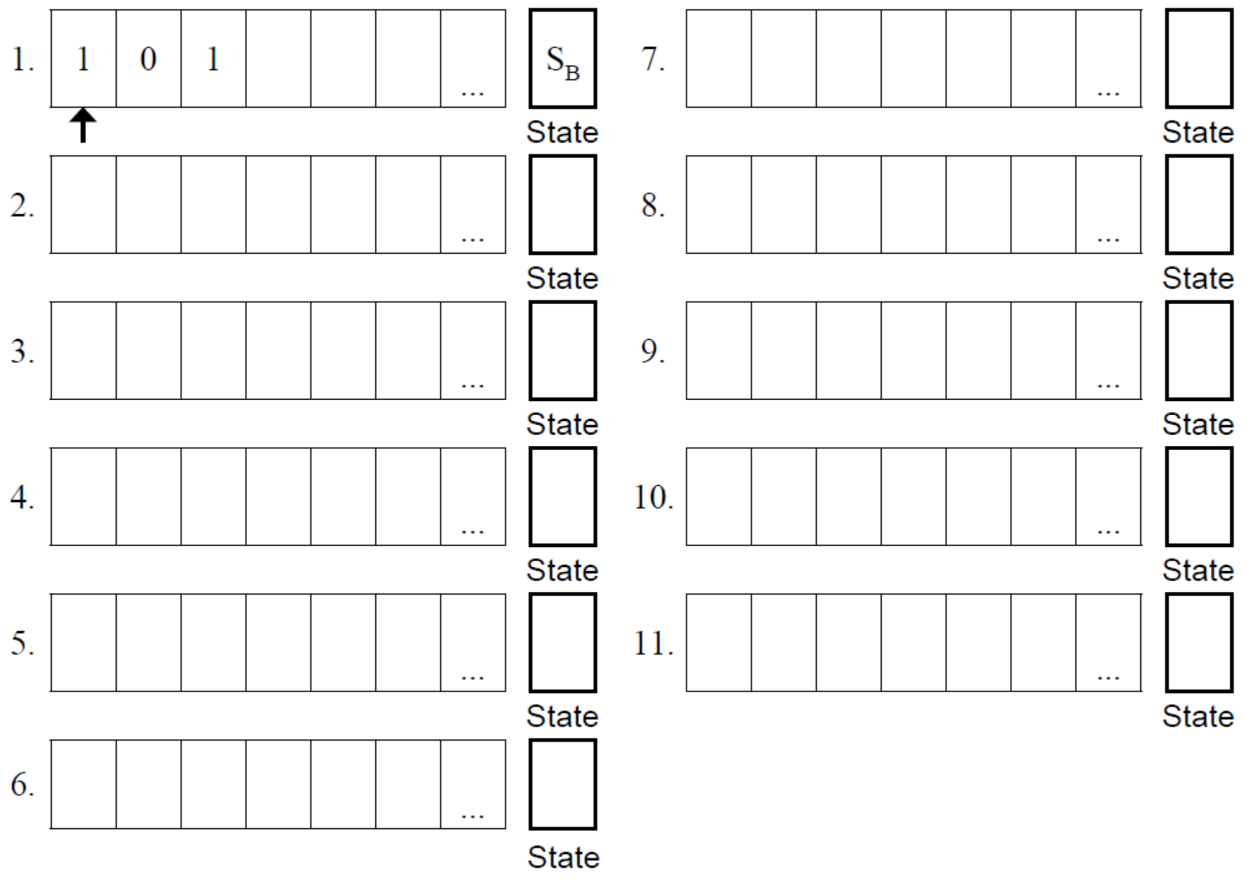
3.1 This Turing machine is carrying out a computation. The machine starts in state S_B with the string 101 on the tape. All other cells contain the blank symbol, □ (not shown). The read/write head is located at the left hand symbol of the string and is indicated with an upward arrow.

Trace the computation of the Turing machine, using the transition function δ.

Show the contents of the tape, the current position of the read/write head and the current state as the input symbols are processed.

The initial configuration of the machine has been completed for you in step 1.

[5 marks]



3.2 The three rules shown below are part of the machine's transition function.

Explain what effect these three rules, taken together, have on the tape, the read/write head and the state of the Turing machine:

$$\begin{aligned} \delta(S_0, 0) &= (S_0, 0, \rightarrow) \\ \delta(S_0, 1) &= (S_0, 1, \rightarrow) \\ \delta(S_0, \square) &= (S_{C0}, \square, \leftarrow) \end{aligned}$$

[2 marks]

3.3 A Universal Turing machine (UTM) is a special type of Turing machine that can be considered to act like an interpreter.

Explain how a UTM can be considered to be an interpreter.

[2 marks]

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.
- Save your program/project/file at regular intervals.

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

One method for converting a decimal number into binary is to repeatedly divide by 2 using integer division. After each division is completed, the remainder is output and the integer result of the division is used as the input to the next iteration of the division process. The process repeats until the result of the division is 0.

Outputting the remainders in the sequence that they are calculated produces the binary digits of the equivalent binary number, but in reverse order.

For example, the decimal number 210 could be converted into binary as shown in **Figure 7**.

Figure 7

210 ÷ 2 =	105	remainder 0
105 ÷ 2 =	52	remainder 1
52 ÷ 2 =	26	remainder 0
26 ÷ 2 =	13	remainder 0
13 ÷ 2 =	6	remainder 1
6 ÷ 2 =	3	remainder 0
3 ÷ 2 =	1	remainder 1
1 ÷ 2 =	0	remainder 1

The sequence 0, 1, 0, 0, 1, 0, 1, 1 which would be output by this process is the reverse of the binary equivalent of 210 which is 11010010.

What you need to do

Task 1
Write a program that will perform the conversion process described above. The program should display a suitable prompt asking the user to input a decimal number to convert and then output the bits of the binary equivalent of the decimal number in reverse order.

Task 2
Improve the program so that the bits are output in the correct order, e.g. for 210 the output would be 11010010.

Task 3
Test the program works by entering the value 210.

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

Evidence that you need to provide
Include the following in your Electronic answer document.

4.1 Your PROGRAM SOURCE CODE after you have completed both **Task 1** and **Task 2**.

If you complete **Task 1** but do not attempt **Task 2** then a maximum of 9 marks will be awarded.

[12 marks]

4.2 SCREEN CAPTURE(S) for the test showing the output of the program when 210 is entered.

The marks for this test will be awarded whether the binary digits are output in reverse order or in the correct order.

[2 marks]

END OF SECTION B

Section ~~B~~ C

You are advised to spend no more than **25 minutes** on this section.

Enter your answers to **Section ~~B~~** C 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.

5 State the name of an identifier for:

5.1 a variable that is used to store a Boolean value.

[1 mark]

5.2 a user-defined subroutine that returns a single Boolean value.

[1 mark]

6.0 The Skeleton Program uses several data structures.

State the identifier of the data structure that stores values of more than one data type.

[1 mark]

7.0 What is the specific purpose of the exception handling construct in the subroutine `SetUpBoard`?

[1 mark]

8.0 This question refers to the subroutines `Game` and `ListPossibleMoves`.

Describe what is contained in the parameter `PlayersPieces` when `ListPossibleMoves` is called for the first time in the subroutine `Game`.

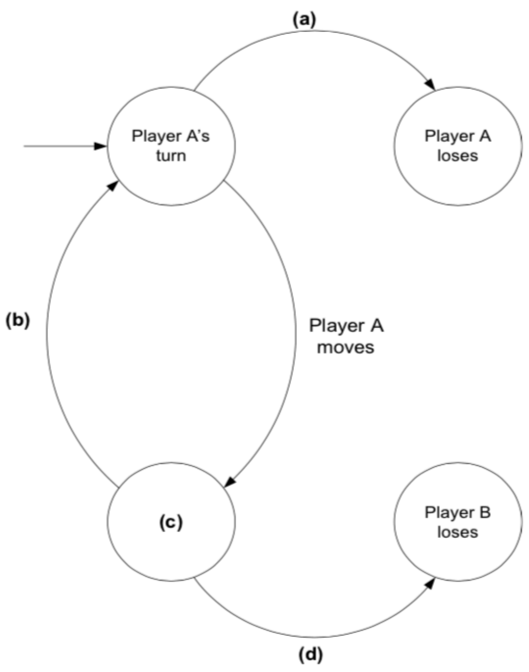
[1 mark]

TURN OVER FOR QUESTION 9

9.0

Figure 3 shows an incomplete state transition diagram for the AQA Board Game. With reference to the game rules complete **Table 3**.

Figure 3



Complete **Table 3** by filling in the unshaded cells with the correct description for **Figure 3**.

Table 3

Label	Description
(a)	
(b)	
(c)	
(d)	

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

[2 marks]

TURN OVER FOR QUESTION 10

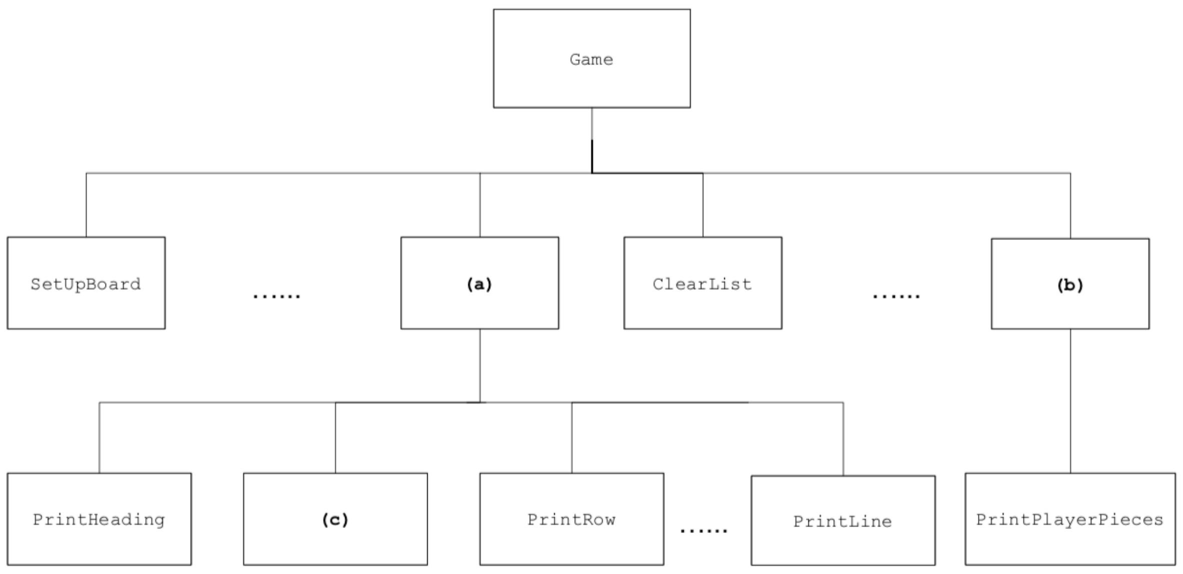
✗

10.0

Figure 4 shows an incomplete hierarchy chart for part of the **Skeleton Program**.

With reference to the **Skeleton Program** and **Figure 4**, answer questions **9.1** to **9.3**.

Figure 4



10.1 What should be written in box **(a)** in **Figure 4**?

[1 mark]

10.2 What should be written in box **(b)** in **Figure 4**?

[1 mark]

10.3 What should be written in box **(c)** in **Figure 4**?

[1 mark]

TURN OVER FOR QUESTION 11

11 This question refers to the data structure `A` defined in the `Game` subroutine.

11.1 Explain the purpose of each of the first two values stored in row 0 in this data structure.

[2 marks]

11.2 Explain why there are 12 rows after row 0 in this data structure.

[1 mark]

11.3 Explain the purpose of the values stored in these 12 rows.

[2 marks]

12 This question refers to the subroutine `CreateNewBoard`.

Describe what the selection structure in this subroutine does.

[3 marks]

13 This question refers to the subroutine `ListPossibleMoves`.

Explain the uses of the variable `NumberOfMoves`.

[2 marks]

14 This question refers to the subroutine `SelectMove`.

Explain what happens in the first `WHILE` loop that terminates when a valid piece has been found.

[5 marks]

Turn over for the next section

Section D

You are advised to spend no more than **60 minutes** on this section.

Enter your answers to SectionD 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.

15 This question refers to the subroutine `SelectMove`. This subroutine makes three calls to the subroutine `DisplayErrorCode` to notify the user of errors. The error codes passed as parameters are to be made more informative for the user.

What you need to do:

Task 1
Modify the subroutine `DisplayErrorCode` so that meaningful error messages are displayed for each type of error explaining the circumstances which caused each error. The error code should also be displayed.

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

- run the program
- enter `Y`
- load `game1.txt`
- enter the string `a4`
- enter the string `a9`
- enter `3`
- enter `4`
- enter `a`
- enter `9`
- enter `3`
- enter `0`

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

15.1 Your PROGRAM SOURCE CODE for the entire subroutine `DisplayErrorCode`.
[3 marks]

15.2 SCREEN CAPTURE(S) showing the requested test including the list of possible moves for Player A.
[1 mark]

TURN OVER FOR QUESTION 16

16

This question refers to the subroutine `ValidJump`. The rules of the game are to be amended. Instead of jumping over their own piece, a player can only jump over an opponent's piece.

What you need to do:

Task 1
Amend the subroutine `ValidJump` so that a jump is only possible if the middle piece belongs to the opponent.

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

- run the program
- enter `Y`
- load `game3.txt`
- enter the string `a5`
- enter `5`
- enter `0`

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

16.1 Your PROGRAM SOURCE CODE for the entire subroutine `ValidJump`.

[2 marks]

16.2 SCREEN CAPTURE(S) showing the requested test including the list of possible moves **before** the jump and your test input, and then the board display **after** the jump.

[1 mark]

Turn over for the next question

17

This question refers to the subroutine `PrintResult`. The rules of the game are to be amended. The game still ends when a player cannot make a move, but the winner is to be determined using a formula that calculates a score for each player. The winner is the player with the **lowest** score. The formula to calculate the score is shown in **Figure 5**.

Figure 5

> Player's score = Number of that player's moves **minus** total number of player's pieces on the board **minus** number of that player's dames **multiplied by** 10
>
> For example, if Player A made 40 moves and has 12 pieces on the board, and 3 of them are dames, then Player A's score is 40 – 12 – (3 × 10) = -2
>
> Note that a dame is also considered to be a piece.

What you need to do:

Task 1
Create a new subroutine `CountNumberOfPieces`. This subroutine is to count the number of pieces a player has on the board.

Task 2
Amend the subroutine `PrintResult` to implement the formula given in **Figure 5** and output the score for each player and display the winner.

You must consider the possibility of a draw.

Task 3
- run the program
- enter `Y`
- load `game4.txt`

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

17.1 Your PROGRAM SOURCE CODE for the entire subroutine `CountNumberOfPieces` and the entire subroutine `PrintResult`.

[9 marks]

17.2 SCREEN CAPTURE(S) showing all the output from the requested test including the board display.

[1 mark]

TURN OVER for Question 18

18

This question will further change the rules of the game.

When a piece is promoted to a dame, the player who the new dame belongs to now chooses **one** of the opponent's pieces. This piece is removed from the board and the dame is placed in the square the removed piece was in.

18.1

State the identifier of the data structure that now needs to be passed as a parameter into the subroutine `MoveDame`.

[1 mark]

What you need to do:

Task 1
Amend the subroutine `MoveDame`.

This subroutine is to:

- ask the player the piece ID of the opponent's piece they want to remove
- check that the piece is an opponent's piece and is on the board
- remove the opponent's piece
- return the coordinates for the new dame.

Task 2
Amend the calls to `MoveDame` from within the subroutine `MovePiece`.

You will need to amend the parameter list of the subroutine heading of `MovePiece` and the call to `MovePiece` from within the subroutine `MakeMove`.

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

- run the program
- enter `Y`
- load `game3.txt`
- move `a2` to row 7, column 0
- take piece `b1`

Task 4

- move `b5` to row 0, column 3
- take piece `a6`

Question 17 continues on the next page

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

18.2 Your PROGRAM SOURCE CODE for the entire subroutine `MoveDame` and the entire subroutine `MakeMove`.

[9 marks]

18.3 SCREEN CAPTURE(S) showing the requested test including the board display after piece `b1` has been taken.

[1 mark]

18.4 SCREEN CAPTURE(S) showing the requested test including the board display after piece `a6` has been taken.

[1 mark]

END OF QUESTIONS