A

AQA

A-level
COMPUTER SCIENCE

Paper 1

7517/1

Time allowed: 2 hours 30 minutes

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 game1.txt.**

You must NOT use a calculator.

[Turn over]

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.

DO NOT TURN OVER UNTIL TOLD TO DO SO

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

The shaded row of TABLE 1, on the opposite page, contains a list of numbers.

A bubble sort algorithm could be used to sort the list of numbers into ascending order.

Complete the unshaded cells of TABLE 1 to show the results of completing THREE passes through the list using a bubble sort algorithm.

You should state the values at the end of each pass.

TABLE 1

	[0]	[1]	[2]	[3]	[4]	[5]
	3	5	8	1	6	4
First pass						
Second pass						
Third pass						

Copy the contents of the unshaded cells in TABLE 1 into the table in your Electronic Answer Document. [3 marks]

[Turn over]

0 2

FIGURE 1 shows a binary tree containing seven nodes. FIGURE 2, on the opposite page, shows how the binary tree in FIGURE 1 could be represented using three one-dimensional arrays: `Data`**,** `Dir1` **and** `Dir2`**.**

FIGURE 1

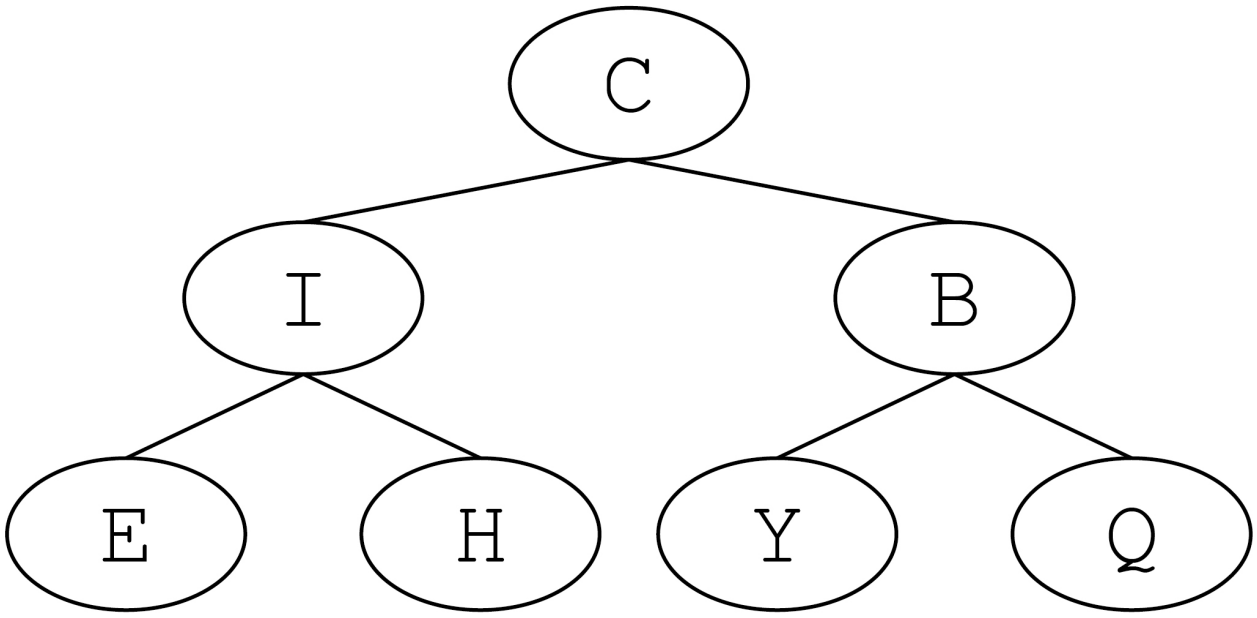


FIGURE 2

Index	Data	Dir1	Dir2
[0]	C	1	4
[1]	I	2	3
[2]	E	-1	-1
[3]	H	-1	-1
[4]	B	5	6
[5]	Y	-1	-1
[6]	Q	-1	-1

0 2 . 1

Define the term binary tree. [2 marks]

0 2 . 2

The output of a post-order traversal algorithm used to print the data item at each node in the binary tree shown in FIGURE 1 would be E, H, I, Y, Q, B, C**.**

State the output that would be produced by an IN-ORDER traversal algorithm. [2 marks]

FIGURE 3 shows pseudo-code for a subroutine called `Traversal` that uses the three arrays from FIGURE 2.

FIGURE 3

```
SUBROUTINE Traversal(StartNode)
  Current ← StartNode
  Pos ← 0
  Stack[Pos] ← Current
  WHILE Pos ≠ -1
    Current ← Stack[Pos]
    Pos ← Pos - 1
    OUTPUT Data[Current]
    IF Dir2[Current] ≠ -1 THEN
      Pos ← Pos + 1
      Stack[Pos] ← Dir2[Current]
    ENDIF
    IF Dir1[Current] ≠ -1 THEN
      Pos ← Pos + 1
      Stack[Pos] ← Dir1[Current]
    ENDIF
  ENDWHILE
ENDSUBROUTINE
```

BLANK PAGE

0 2 . 3

Complete the unshaded cells in TABLE 2 to show the result of the subroutine call `Traversal(0)`

TABLE 2

		Stack				
Current	Pos	[0]	[1]	[2]	[3]	Output

Copy the contents of the unshaded cells in TABLE 2 into the table in your Electronic Answer Document. [7 marks]

[Turn over]

FIGURE 3 (REPEATED)

```
SUBROUTINE Traversal(StartNode)
  Current ← StartNode
  Pos ← 0
  Stack[Pos] ← Current
  WHILE Pos ≠ -1
    Current ← Stack[Pos]
    Pos ← Pos - 1
    OUTPUT Data[Current]
    IF Dir2[Current] ≠ -1 THEN
      Pos ← Pos + 1
      Stack[Pos] ← Dir2[Current]
    ENDIF
    IF Dir1[Current] ≠ -1 THEN
      Pos ← Pos + 1
      Stack[Pos] ← Dir1[Current]
    ENDIF
  ENDWHILE
ENDSUBROUTINE
```

0 2 . 4

The subroutine shown in FIGURE 3 could have been written so that it used recursion instead of iteration.

Explain what is meant by a recursive subroutine.
[1 mark]

0 2 . 5

Explain what is meant by a base case for a recursive subroutine. [1 mark]

0 2 . 6

If the subroutine shown in FIGURE 3 had been written using recursion, a stack frame would have been stored each time a recursive subroutine call was made.

State TWO components of a stack frame. [2 marks]

[Turn over]

0 3

Explain what is meant by procedural decomposition. [3 marks]

0 4

Describe the steps involved in adding a record to a hash table. [5 marks]

0 5 . 1

State TWO advantages of using Reverse Polish Notation (RPN) instead of infix notation to represent an expression. [2 marks]

0 5 . 2

Describe how a single stack could be used to evaluate an RPN expression. [4 marks]

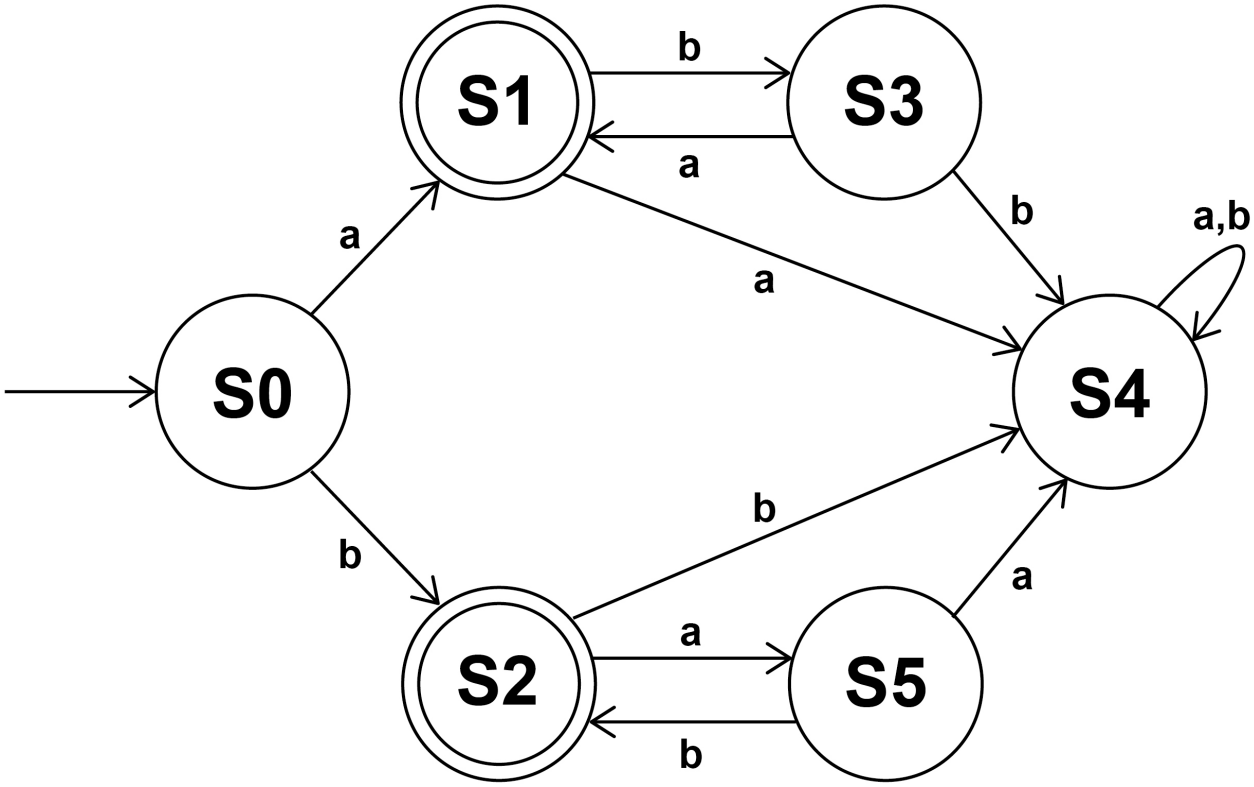
BLANK PAGE

[Turn over]

0 6

The state transition diagram for a finite state machine (FSM) is shown in FIGURE 4.

FIGURE 4



0 6 . 1

An FSM can also be represented as a state transition table.

Complete the state transition table, TABLE 3, so that it represents the parts of the state transition diagram shown in FIGURE 4 that involve state S2.

TABLE 3

Current state	Input	New state

Copy the contents of the unshaded cells in TABLE 3 into the table in your Electronic Answer Document. [2 marks]

0 6 . 2

Regular expressions can be used to recognise the same strings as FSMs without output.

Write a regular expression that will recognise the same set of strings that are accepted by the FSM shown in FIGURE 4. [3 marks]

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 7

A Harshad number is a positive integer which is exactly divisible by the sum of its digits. The first twelve Harshad numbers are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12 and 18

- **36 is a Harshad number. The digits of 36 are 3 and 6; the sum of these digits is 9 as 3 + 6 = 9 and 36 is exactly divisible by 9 (36 ÷ 9 = 4)**
- **300 is a Harshad number. The digits of 300 are 3, 0 and 0; the sum of these digits is 3 as 3 + 0 + 0 = 3 and 300 is exactly divisible by 3 (300 ÷ 3 = 100)**

- **15 is not a Harshad number. The digits of 15 are 1 and 5; the sum of these digits is 6 as 1 + 5 = 6 and 15 is not exactly divisible by 6**

Write a program that asks the user to enter a number, *n*, and will then calculate and display the *n*th Harshad number.

EXAMPLE

If the user enters the number 12 then the program should calculate and display the twelfth Harshad number. The twelfth Harshad number is 18

You may assume that the number that the user enters will be a positive integer.

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

0 7 . 1

Your PROGRAM SOURCE CODE. [12 marks]

0 7 . 2

SCREEN CAPTURE(S) showing the result of testing the program by entering the number `600` **[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 8

To specify which tile to use when entering a command, a player enters the index of the tile in the `Tiles` list. Instead of using this index the player could have been asked to enter the x, y and z coordinates for the tile.

FIGURE 5, on the opposite page, shows the coordinates and indices for each of the tiles in the default game.

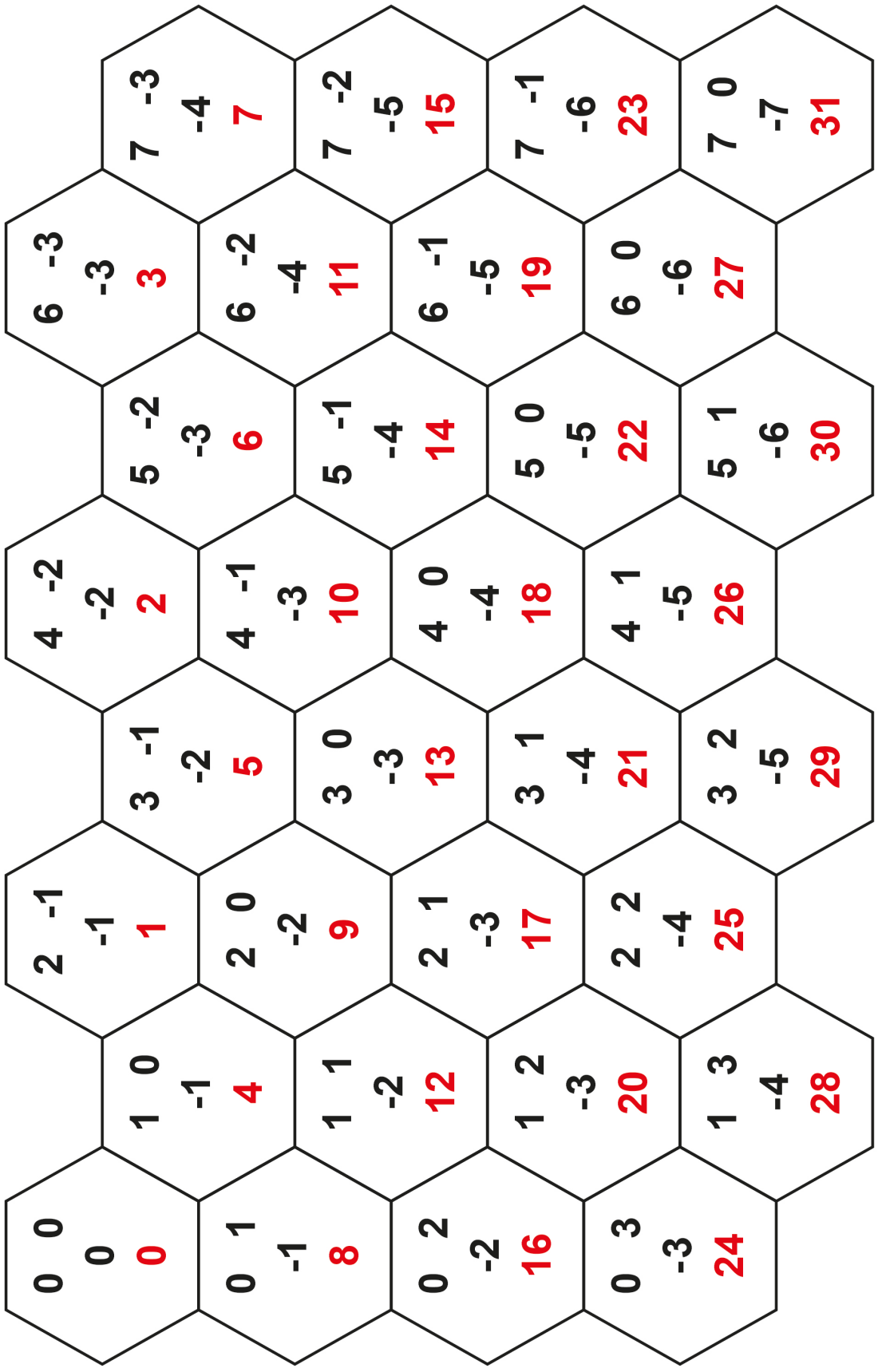


FIGURE 5

[Turn over]

BLANK PAGE

0 8 . 1

In the default game the tile with an index of 18 is in a straight line with the tile which has an index of 0. It is also in a straight line with the tile which has an index of 7 and in a straight line with the tile which has an index of 2.

Explain how you can tell from the coordinates of two tiles if they are in a straight line with each other.
[1 mark]

0 8 . 2

FIGURE 6 shows an incomplete calculation.

FIGURE 6

$$\left(\frac{\texttt{GridSize}}{2}\right) \times (\text{difference between } z \text{ and } y \text{ coordinates}) + \text{(A)}$$

What should (A) be replaced with so that the index of a tile in `Tiles` can be calculated from the coordinates of the tile? [2 marks]

[Turn over]

0 8 . 3

Describe the modifications that would need to be made to the `CheckMoveCommandFormat` subroutine so that the `move` command could use coordinates instead of indices for the two tiles. [2 marks]

0 9

This question is about the different types of piece in the game.

0 9 . 1

Explain why the `FuelCostOfMove` attribute in the `Piece` class could NOT have been a private attribute. [1 mark]

0 9 . 2

Describe the circumstances when there could be no baron pieces on the grid but more commands would still have to be entered. [2 marks]

0 9 . 3

In object-oriented programming, what is meant by polymorphism? [1 mark]

1 0

The `LoadGame` **subroutine uses exception handling to prevent potential runtime errors. An example of an event that could cause a runtime error when executing the** `LoadGame` **subroutine would be trying to open a file that does not exist.**

1 0 . 1

Describe another event that could cause a runtime error when executing the `LoadGame` **subroutine. [1 mark]**

1 0 . 2

State the identifier of another subroutine that uses exception handling. [1 mark]

1 1

This question is about the `GetDistanceToTileT` **subroutine in the** `Tile` **class.**

Explain how this subroutine calculates the distance between two tiles. [2 marks]

SECTION D

You are advised to spend no more than 70 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 to make programming changes to it.

1 2

This question refers to the subroutine `DestroyPiecesAndCountVPs` **in the** `HexGrid` **class.**

The victory point scoring system for the game is to be changed so that at the end of each turn both players gain additional victory points based on how many LESS pieces they have on the board.

WHAT YOU NEED TO DO

TASK 1

Modify the subroutine `DestroyPiecesAndCountVPs` **so that if a piece has NOT been destroyed it checks to see if it is a LESS piece. If it is a LESS piece the number of victory points**

awarded to the player to whom that piece belongs should be increased by one.

TASK 2

Test that the changes you have made work:

- **run the Skeleton Program**
- **choose to load a game**
- **enter the filename game1.txt**
- **keep pressing the Enter key until BOTH players have had a turn and the grid has been shown at the start of Player One's second turn.**

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

12.1

Your PROGRAM SOURCE CODE for the amended subroutine `DestroyPiecesAndCountVPs`**. [5 marks]**

12.2

SCREEN CAPTURE(S) showing the results of the requested test. [1 mark]

[Turn over]

1 3

This question adds a new type of piece to the game, a ranger, that moves differently to the other pieces.

A ranger can move in the same way as a standard (serf) piece but can also move directly to any available forest tile in the grid if the ranger is currently in a forest tile. The cost of making this type of move is one fuel.

WHAT YOU NEED TO DO

TASK 1

Create a new class called `RangerPiece` **that is a subclass of the** `Piece` **class. The constructor for this new class should make a call to the constructor of the** `Piece` **class and then set the value of** `PieceType` **to** `R`**.**

TASK 2

Create a subroutine `CheckMoveIsValid` **in the new** `RangerPiece` **class that overrides the subroutine from the base class and allows a ranger piece to move in the way described.**

TASK 3

Modify the subroutine `AddPiece` **in the** `HexGrid` **class so that it creates a new** `RangerPiece` **if** `TypeOfPiece` **is** `Ranger`**.**

TASK 4

Modify the subroutine `SetupDefaultGame` **so that Player One has a ranger piece in tile 8 instead of a serf piece.**

TASK 5

Test that the changes you have made work:

- **run the Skeleton Program**
- **choose the default game**
- **enter the command** `move 8 12`
- **enter the command** `move 12 2`
- **enter the command** `move 2 3`
- **then press the Enter key so that the grid is displayed showing the results of these commands.**

[Turn over]

BLANK PAGE

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

1 3 . 1

Your PROGRAM SOURCE CODE for the new class `RangerPiece` **and the amended subroutine** `AddPiece`**. [7 marks]**

1 3 . 2

SCREEN CAPTURE(S) showing the requested test. [1 mark]

[Turn over]

1 4

This question extends the Skeleton Program by adding a new command to the game that allows a player to burn lumber to turn it into fuel.

If a player uses the `burn` **command when they have lumber in their supply a random integer between one and the amount of lumber they have in their supply is generated. The amount of lumber in their supply is decreased by this random integer with the amount of fuel in their supply being increased by the same amount.**

If a player uses the `burn` **command when they do not have any lumber in their supply the message** `Cannot burn lumber` **is displayed and no fuel is created.**

WHAT YOU NEED TO DO

Task 1

Modify the `CheckCommandIsValid` **subroutine so that it returns** `True` **if the command was** `burn`**.**

TASK 2

Modify the `ExecuteCommand` **subroutine in the** `HexGrid` **class so that when a player chooses the** `burn` **command it returns the string** `Cannot burn lumber` **if the player does not have any lumber in their supply.**

If the player does have lumber in their supply it:

- **generates a random integer between one and the amount of lumber in the player's supply**
- **decreases the amount of lumber in the player's supply by the random number generated**
- **increases the amount of fuel in the player's supply by the random number generated**
- **returns the string** `Command executed`**.**

TASK 3

Test that the changes you have made work:

- **run the Skeleton Program**
- **choose the default game**
- **enter the command** `burn`
- **press the Enter key twice.**

[Turn over]

BLANK PAGE

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

1 4 . 1

Your PROGRAM SOURCE CODE for the amended subroutines `CheckCommandIsValid` **and** `ExecuteCommand`**. [8 marks]**

1 4 . 2

SCREEN CAPTURE(S) showing the requested test. [1 mark]

[Turn over]

1 5

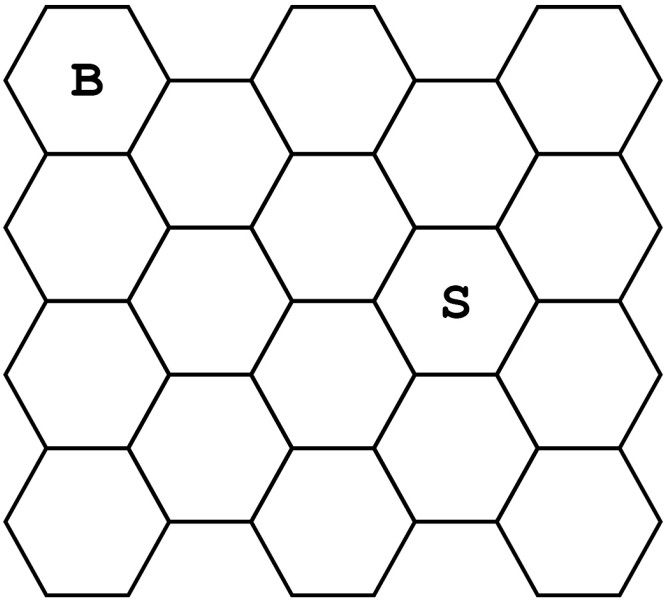
A new feature called 'fog of war' is to be added to the Skeleton Program. Fog of war means that each player will only be shown the location of a piece belonging to their opponent if that piece is near one of their own pieces. A piece is near another piece if it is two or fewer cells away on the grid.

The player will still be shown the terrain that is in all the tiles.

FIGURES 7 to 11 show an example of how the fog of war feature should work.

FIGURE 7 shows the current positions of Player One's pieces.

FIGURE 7



In FIGURE 8, on the opposite page the shaded tiles are those that are two or fewer cells away from Player One's baron piece.

In FIGURE 9 the shaded tiles are those that are two or fewer cells away from Player One's serf piece.

FIGURE 8

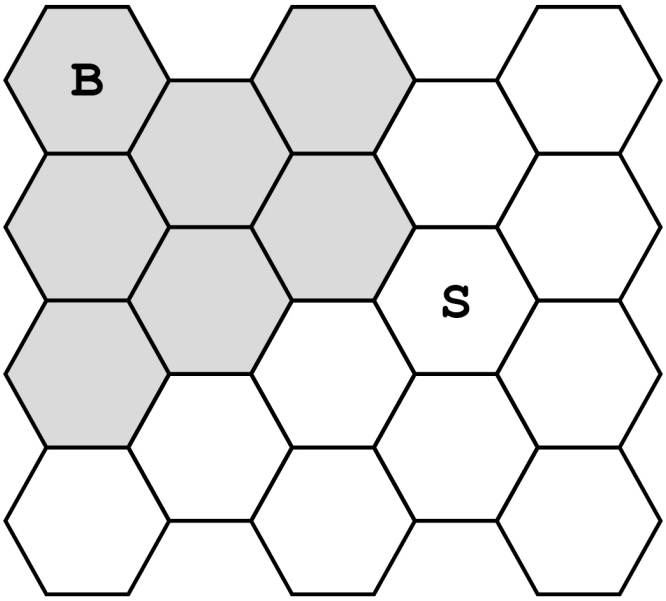


FIGURE 9

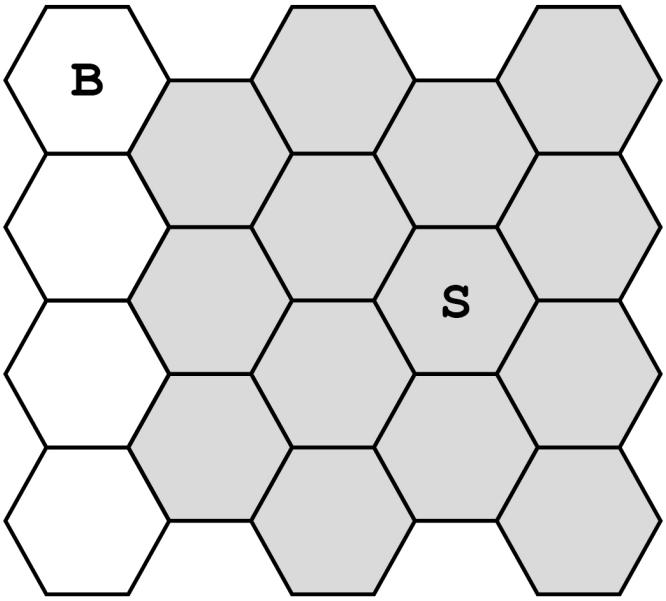


FIGURE 10 shows the positions of Player One's pieces and the positions of Player Two's pieces.

FIGURE 11, on the opposite page, shows what Player One should see when they are shown the grid when the fog of war feature has been implemented.

- **Player Two's baron piece can be seen because it is within two cells of Player One's serf piece.**
- **Player Two's LESS piece can be seen because it is within two cells of both Player One's serf piece and Player One's baron piece.**
- **Player Two's serf piece can be seen because it is within two cells of Player One's baron piece.**
- **Player Two's PBDS piece cannot be seen because it is not within two cells of any of Player One's pieces.**

FIGURE 10

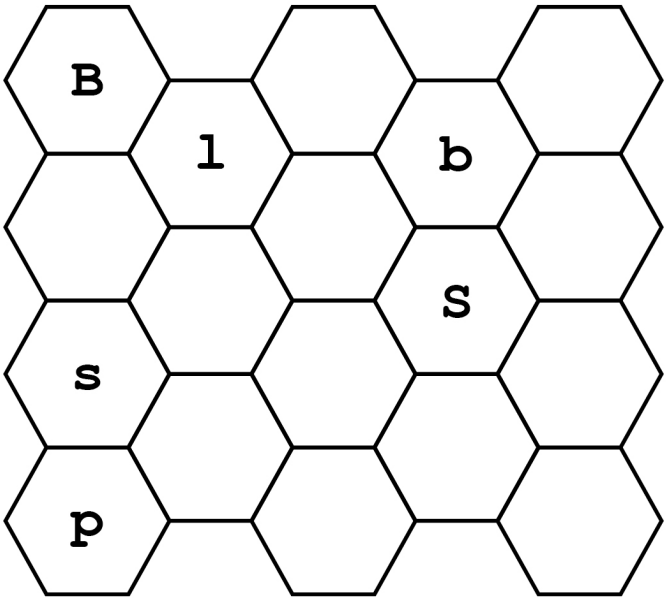
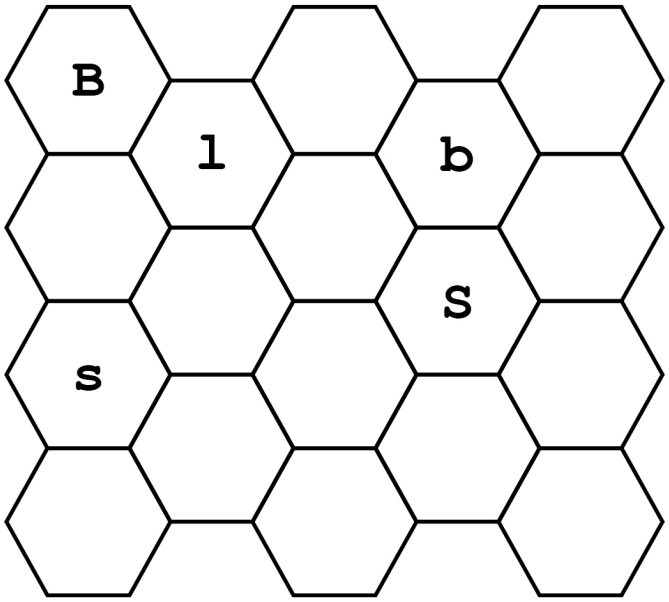


FIGURE 11



WHAT YOU NEED TO DO

TASK 1

Create a new subroutine `GetFogOfWar` **in the** `HexGrid` **class.**

The new subroutine should take the index of a tile in `Tiles` **and return** `False` **if the player whose turn it is has any piece which is two or fewer cells away from that tile as the tile is not hidden because of fog of war. Otherwise, it should return** `True` **as this tile is hidden because of fog of war.**

TASK 2

Modify the `GetPieceTypeInTile` **subroutine in the** `HexGrid` **class so that it uses the** `GetFogOfWar` **subroutine to determine if this tile is affected by fog of war.**

If the contents of this tile would be hidden because of fog of war, it should return the string consisting of a single space character.

If the contents would not be hidden because of fog of war, the existing functionality of the `GetPieceTypeInTile` **subroutine should not be changed.**

TASK 3

Test that the changes you have made work:

- **run the Skeleton Program**
- **choose to load a game**
- **enter the filename game1.txt**
- **keep pressing the Enter key until the grid has been shown at the start of Player Two's first turn.**

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

1 5 . 1

Your PROGRAM SOURCE CODE for the amended subroutine `GetPieceTypeInTile` **and for the new subroutine** `GetFogOfWar`**. [13 marks]**

1 5 . 2

SCREEN CAPTURE(S) showing the requested test. The screen capture should show all the output displayed by the program. [1 mark]

END OF QUESTIONS

BLANK PAGE