A

AQA

A-level
COMPUTER SCIENCE
Paper 1
7517/1

Time allowed: 2 hours 30 minutes

[Turn over]

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 Files game1.txt and locks.txt**
- **an insert**

You must NOT use a calculator.

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.**

[Turn over]

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 – 40 minutes
SECTION B – 20 minutes
SECTION C – 20 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 40 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

Big-O notation is used to express the time complexity of an algorithm. TABLE 1 contains a list of algorithms.

State the Big-O time complexity of each of these algorithms. The first row has been completed for you.

TABLE 1

Algorithm	Time complexity
Binary tree search	O(log n)
Bubble sort	
Linear search	
Merge sort	

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

[Turn over]

0 2

A queue data structure can be implemented as a static data structure using an array.

0 2 . 1

Describe the method that would need to be followed to attempt to remove an item from a circular queue implemented as a static data structure using an array.

Your method should deal appropriately with any issues which could arise.
[4 marks]

0 2 . 2

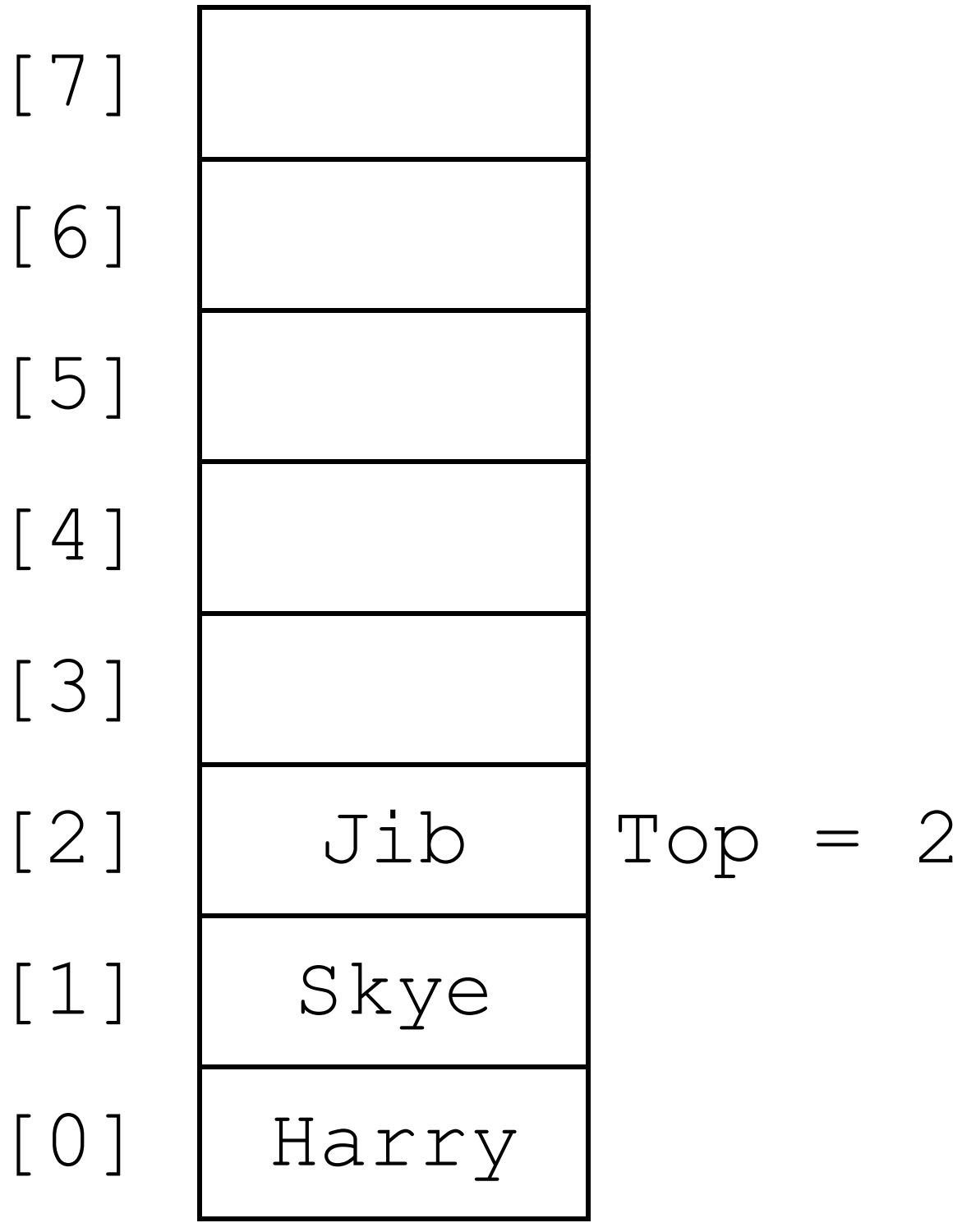
Describe THREE differences between dynamic and static data structures.
[3 marks]

BLANK PAGE

[Turn over]

FIGURE 1 shows data that has been stored in a stack implemented using an array S**.**

FIGURE 1



0 2 . 3

What value will be returned by applying the `peek` operation to `S`? [1 mark]

0 2 . 4

What value will be returned by applying the `pop` operation to `S`? [1 mark]

0 2 . 5

Explain how a single stack can be used to reverse the order of the items in a queue. [2 marks]

0 3

FIGURE 2 shows a logic puzzle.

FIGURE 2

Which one of these six statements is correct?

Statement 1: All of the statements below are correct.

Statement 2: None of the statements below are correct.

Statement 3: All of the statements above are correct.

Statement 4: Exactly one of the statements above is correct.

Statement 5: None of the statements above are correct.

Statement 6: None of the statements above are correct.

0 3 . 1

Explain why Statement 1 is not correct. [1 mark]

0 3 . 2

Which one of the six statements in FIGURE 2 is correct? [1 mark]

0 3 . 3

For TWO statements other than Statement 1 and your answer to Question 03.2, explain why those statements are not correct. [2 marks]

0 4 . 1

For each of the statements in TABLE 2, complete each row to indicate if the statement is true or false for Dijkstra's algorithm.

TABLE 2

	True or False?
Calculates the shortest path between a node and other nodes in a graph.	
Can be used to prove that the Halting Problem cannot be solved.	
Can be used with both directed and undirected graphs.	
Can be used with both weighted and unweighted graphs.	

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

FIGURE 3, on page 2 of the insert, shows a subroutine represented using pseudo-code. The subroutine makes use of an array `Visited` **and an array** `ConnectedNodes` **that stores a graph represented as an adjacency list.**

0 4 . 2

The subroutine `G` **uses recursion.**

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

FIGURE 4, provided on page 3 of the insert, shows a subroutine represented using pseudo-code. The subroutine makes use of the array `Visited`**.**

FIGURE 5, provided on page 3 of the insert, shows a subroutine represented using pseudo-code. The subroutine makes use of the subroutine `G` **shown in FIGURE 3, the subroutine** `F` **shown in FIGURE 4 and the array** `Visited`**.**

FIGURE 6, provided on page 4 of the insert, shows a graph consisting of three nodes, the contents of the array `ConnectedNodes` **when it is used to represent this graph, and the contents of the array** `Visited` **after the subroutine call** `G(0, -1)`**.**

0 4 . 3

Complete the unshaded cells in TABLE 3 to show the result of the subroutine call `F()` when it is applied using the graph shown in FIGURE 6.

TABLE 3

`Count`	**Value returned**

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

FIGURE 7, provided on page 5 of the insert, shows a graph consisting of four nodes and the contents of the array `ConnectedNodes` when it is used to represent this graph.

0 4 . 4

Complete the unshaded cells in TABLE 4 to show how the graph in FIGURE 7 would be represented as an adjacency matrix.

TABLE 4

	0	1	2	3
0				
1				
2				
3				

Copy the contents of the unshaded cells in TABLE 4 into the table in your Electronic Answer Document. [1 mark]

0 4 . 5

Complete the unshaded cells in TABLE 5, on the opposite page, to show the result of the subroutine call `G(0, -1)` **on the graph shown in FIGURE 7. Some parts of the table, including the initial values in the** `Visited` **array, have been completed for you.**

Copy the contents of the unshaded cells in TABLE 5, on the opposite page, into the table in your Electronic Answer Document. [6 marks]

TABLE 5

			Visited				
Subroutine call	**V**	**P**	**[0]**	**[1]**	**[2]**	**[3]**	**N**
			False	False	False	False	
G(0, -1)							
Final value returned:							

[Turn over]

BLANK PAGE

0 4 . 6

What is the purpose of the subroutine `G`**?**
[1 mark]

0 4 . 7

State the type of graph traversal used in subroutine `G`**. [1 mark]**

0 4 . 8

If the graph represented by `ConnectedNodes` **is undirected, what can you determine about the graph when a value of** `True` **is returned by subroutine** `E`**? [1 mark]**

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	5

Write a program that asks the user to enter a string. It should then change the order of the vowels in the string and display the result.

If there are *n* vowels in the string, the 1st vowel in the string should swap with the *n*th vowel in the string, the 2nd vowel in the string should swap with the (*n-1*)th vowel in the string, and so on.

The letters `a`**,** `e`**,** `i`**,** `o` **and** `u` **are the only vowels.**

[Turn over]

EXAMPLES

If the user enters the string `horse` **then the program should display the string** `herso`**.**

If the user enters the string `goose` **then the program should display the string** `geoso`**.**

If the user enters the string `pinkfairyarmadillo` **then the program should display the string** `ponkfiaryarmidalli`**.**

If the user enters the string `nakedmolerat` **then the program should display the string** `nakedmolerat`**.**

If the user enters the string `lynx` **then the program should display the string** `lynx`**.**

If the user enters the string `pig` **then the program should display the string** `pig`**.**

You may assume the string that the user enters will only contain lowercase letters.

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

0 5 . 1

Your PROGRAM SOURCE CODE.
[12 marks]

0 5 . 2

SCREEN CAPTURE(S) showing the results of three tests of the program by entering the strings `persepolis`, `darius` **and** `xerxes`. **[1 mark]**

BLANK PAGE

SECTION C

You are advised to spend no more than 20 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.

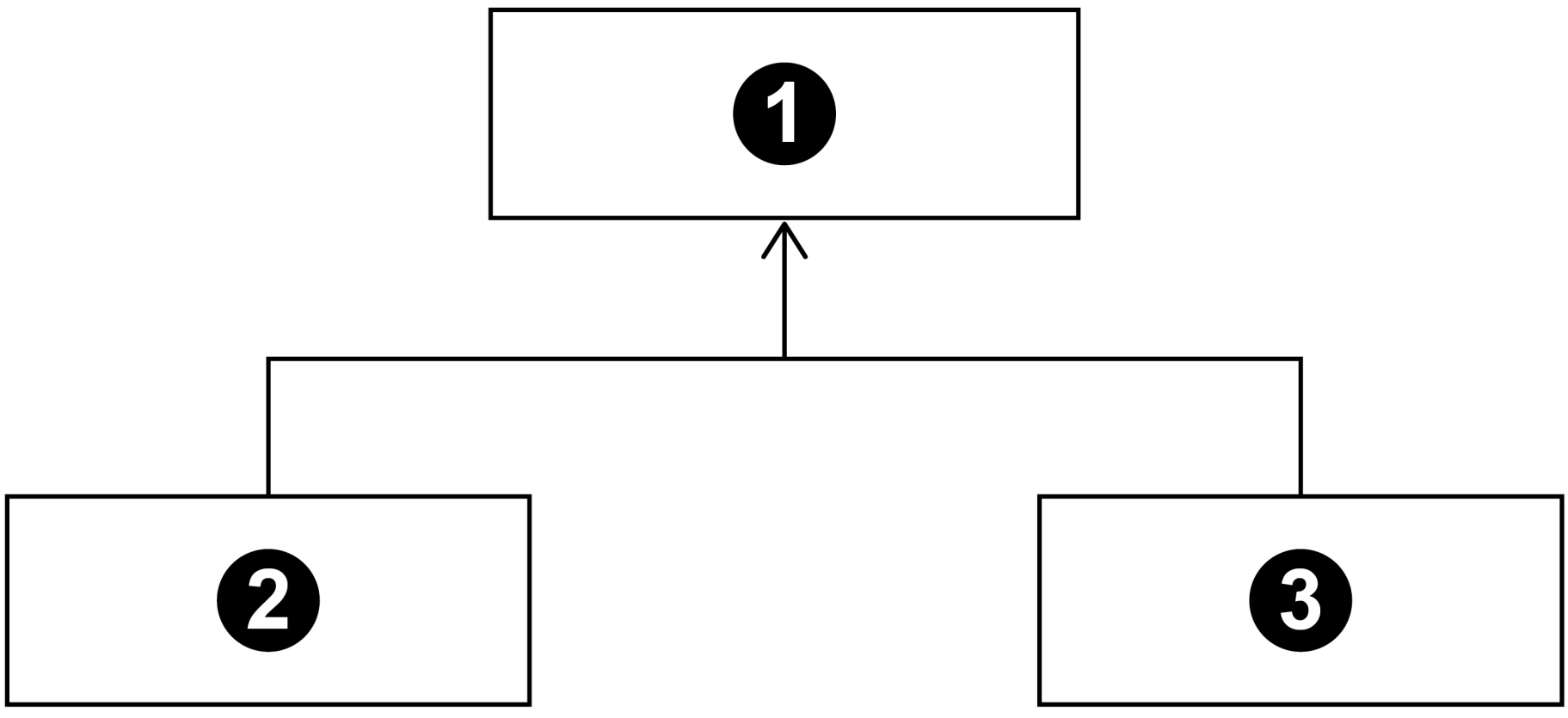
Refer EITHER to the PRELIMINARY MATERIAL issued with this question paper OR your electronic copy.

[Turn over]

0 6

An incomplete class diagram of the user-defined classes for part of the Skeleton Program is shown in FIGURE 8.

FIGURE 8



0 6 . 1

State the type of relationship the diagram in FIGURE 8 shows between the class indicated by ❶ and the class indicated by ❷. [1 mark]

0 6 . 2

State the identifier of the class indicated by ❶ in FIGURE 8. [1 mark]

0 6 . 3

Explain the difference between an attribute that has a public specifier and an attribute that has a protected specifier. [2 marks]

0 6 . 4

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

0 7

This question is about the `CardCollection` **class.**

0 7 . 1

FIGURE 9 shows a pseudo-code version of part of the `Shuffle` **subroutine.**

FIGURE 10 shows an alternative, incorrect version of the same part of the `Shuffle` **subroutine.**

FIGURE 9

```
TempCard ← Cards[RNo1]
Cards[RNo1] ← Cards[RNo2]
Cards[RNo2] ← TempCard
```

FIGURE 10

```
Cards[RNo1] ← Cards[RNo2]
Cards[RNo2] ← Cards[RNo1]
```

Explain why the `Shuffle` subroutine would not work if it used the method shown in FIGURE 10 instead of the method shown in FIGURE 9. [1 mark]

[Turn over]

The `CardCollection` **class uses a list to store the cards.**

0 7 . 2

State ONE reason why a set could NOT have been used instead of a list.
[1 mark]

0 7 . 3

A hash table could have been used instead of a list.

Describe how a card would be added to a hash table. [3 marks]

0 7 . 4

A hash table can be used to implement a dictionary data structure.

Explain why a hash table is a suitable choice. [1 mark]

[Turn over]

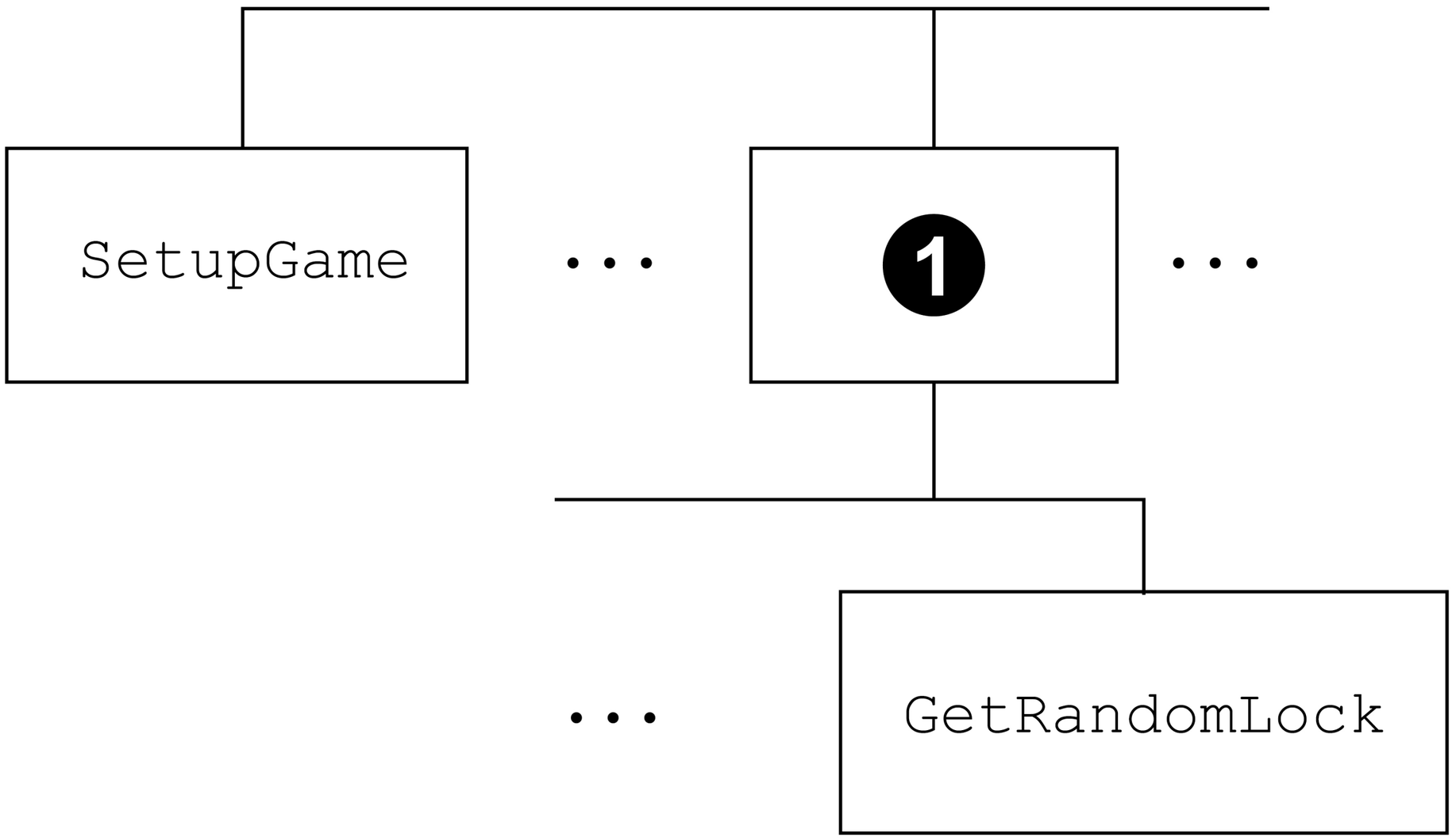
0 8

An incomplete hierarchy diagram of the user-defined subroutines in part of the Skeleton Program is shown in FIGURE 11, on the opposite page.

0 8 . 1

Which identifier should replace ❶? [1 mark]

FIGURE 11



[Turn over]

0 8 . 2

The value `5` **is used in the subroutine** `SetupGame`**. It would be better to use a named constant with an identifier that describes the purpose of the constant.**

Suggest a suitable identifier for the named constant.
[1 mark]

0 8 . 3

State one actual data value that will always be in the stack frame added to the stack when the subroutine `LoadGame` **is called from the** `SetupGame` **subroutine. [1 mark]**

0 8 . 4

Explain why the call to the subroutine `AddDifficultyCardsToDeck` is after, not before, the iteration structure in the `SetupGame` subroutine. [1 mark]

[Turn over]

0	9

How many subroutines in the Skeleton Program access external data files? [1 mark]

1	0

The user must input a `D` or `P` to select Discard or Play during a game.

Write a regular expression that would match the character `D` or `P`.

You should NOT make any changes to the Skeleton Program to answer this question. [1 mark]

BLANK PAGE

[Turn over]

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	1

This question refers to the subroutine `GetDiscardOrPlayChoice` **in the** `Breakthrough` **class.**

The program is to be changed so that it checks that the choice entered after the player has chosen to use a card is valid. `D` **and** `P` **are the only valid choices. The program should keep getting the player to enter a value until a valid choice has been made.**

WHAT YOU NEED TO DO

TASK 1

Modify the subroutine `GetDiscardOrPlayChoice` **so it checks that the value entered by the player is valid. An appropriate error message should be displayed if an invalid value is entered and the user should be made to enter another value.**

[Turn over]

TASK 2

Test that the changes you have made work:

- **run the Skeleton Program**
- **play a new game**
- **enter** `U`
- **enter** `2`
- **enter** `L`
- **enter** `D`

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

1 1 . 1

Your PROGRAM SOURCE CODE for the amended subroutine `GetDiscardOrPlayChoice`**. [4 marks]**

1 1 . 2

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

1	2

This question extends the Skeleton Program so that it displays some information about the contents of the deck to the player.

The program needs to be modified so that it displays messages telling the player how many cards are in the deck and how many tool cards there are in the deck. A tool card is a pick, file, or key.

WHAT YOU NEED TO DO

TASK 1

Modify the `PlayGame` **subroutine in the** `Breakthrough` **class so that a message is displayed telling the player how many cards there are in the deck.**

This message should be displayed after the message telling the player what their

current score is and before the contents of the player's hand are displayed.

TASK 2

Create a subroutine `GetNumberOfToolCards` **in the** `CardCollection` **class that calculates how many tool cards there are in the** `Cards` **data structure. It should return the calculated value to the calling routine.**

TASK 3

Modify the `PlayGame` **subroutine in the** `Breakthrough` **class so that a message is displayed telling the player how many tool cards there are in** `Deck`**. This message should use the value returned by calling the** `GetNumberOfToolCards` **subroutine for** `Deck`**.**

[Turn over]

This message should be displayed after the message telling the player what their current score is and before the contents of the player's hand are displayed.

TASK 4

Test that the changes you have made work:

- **run the Skeleton Program**
- **play a new game**
- **enter** `U`
- **enter** `2`
- **enter** `D`

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

1 2 . 1

Your PROGRAM SOURCE CODE for the new subroutine `GetNumberOfToolCards` **and the amended** `PlayGame` **subroutine.**
[8 marks]

1 2 . 2

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

1	3

This question extends the Skeleton Program by allowing a player to use a blasting cap to complete any challenge on the current lock.

The player will only be able to do this once per game.

When the player chooses to use the blasting cap, the program should check if the player has already used the blasting cap. If they have not used the blasting cap, the program should:

- **change the status of the blasting cap to show that it has been used**
- **ask the player to enter the position of the challenge on the current lock they would like to use the blasting cap on (1 for the first challenge, 2 for the second challenge, etc)**

- check:
  - that the position entered is less than or equal to the number of challenges on the current lock
  - that the challenge in that position has not already been met.
- if both these checks are passed:
  - mark the challenge as being met
  - display a message saying the blasting cap was used successfully
  - display the details for the current lock.
- if either of these checks fail, then the blasting cap has been wasted and the game continues.

If the player tries to use the blasting cap when they have already used it then nothing changes and the game continues.

[Turn over]

WHAT YOU NEED TO DO

TASK 1

Modify the `GetChoice` **subroutine in the** `Breakthrough` **class so the message displayed shows the blasting cap option.**

TASK 2

Modify the `PlayGame` **subroutine in the** `Breakthrough` **class so that there is a blasting cap that works in the way described.**

TASK 3

Test that the changes you have made work:

- **run the Skeleton Program**
- **enter** `L`
- **choose to use a blasting cap**

- **choose to complete the third challenge on the current lock**
- **choose to use a blasting cap.**

EVIDENCE THAT YOU NEED TO PROVIDE

Include the following evidence in your Electronic Answer Document.

13.1

Your PROGRAM SOURCE CODE for the amended subroutines `GetChoice` **and** `PlayGame`**. [9 marks]**

13.2

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

1	4

This question adds a new type of difficulty card to the game, a trap.

When the player draws a trap card from the deck, it works as follows:

- **if no challenges from the current lock have been met, the trap card works in the same way as a difficulty card and the player's choice to lose a key or discard five cards is executed**
- **if one or more challenges from the current lock have been met, one of the met challenges is randomly chosen and has its status changed so it is no longer met. This happens instead of executing the player's choice to lose a key or discard five cards from the deck.**

Trap cards are used in the game instead of difficulty cards when the player chooses to load a game from a file. They

are NOT used when the player chooses to play a new game.

WHAT YOU NEED TO DO

TASK 1

Create a new class called `TrapCard` **that is a subclass of the** `DifficultyCard` **class.**

The constructor for this new class should set the value of `CardNumber` **to the value of the constructor's parameter and set the value of** `CardType` **to** `Trp`**.**

Create a subroutine `Process` **in the** `TrapCard` **class that overrides the subroutine from the** `DifficultyCard` **class and allows a trap card to work in the way described.**

[Turn over]

TASK 2

Modify the `SetupCardCollectionFromGameFile` **subroutine in the** `Breakthrough` **class so that it creates** `TrapCards` **instead of** `DifficultyCards`**.**

TASK 3

Modify the `GetCardFromDeck` **subroutine in the** `Breakthrough` **class so that if the top card of the deck has a** `CardType` **of** `Trp`**, that card is treated in the same way as if the top card of the deck had a** `CardType` **of** `Dif`**.**

When the top card of the deck has a `CardType` **of** `Trp`**, the message** `"Trap!"` **should be displayed.**

TASK 4

Test that the changes you have made work:

- **run the Skeleton Program**
- **enter** `L`
- **enter** `U`
- **enter** `1`
- **enter** `P`
- **enter** `U`
- **enter** `1`
- **enter** `P`
- **enter** `D`

[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 subroutine `GetCardFromDeck`**, the amended subroutine** `SetupCardCollectionFromGameFile` **and the new class** `TrapCard`**.**
[12 marks]

1 4 . 2

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

END OF QUESTIONS

BLANK PAGE

IB/M/CD/Jun22/7517/1/E3

2 2 6 A 7 5 1 7 / 1