OXFORD AQA
INTERNATIONAL QUALIFICATIONS

The international exam board *that puts fairness first*

OxfordAQA International GCSE

Computer Science (9210)

Example responses (9210/1)

For teaching from September 2017 onwards
For International GCSE exams from May/June 2019 onwards

OxfordAQA is a partnership between AQA, the UK's largest awarding body, and Oxford University Press, a department of the University of Oxford. We offer globally relevant international GCSEs, AS and A-levels to schools across Europe, Africa and Asia.

Contents

The below content table is interactive. You can press the control button click on the title of the question to go directly to that page.

Introduction

This guide includes students' responses to questions from the June 2022 International GCSE paper 9210/1.

Assessment Objectives

The exams will measure how students have achieved the following assessment objectives:

- AO1: Demonstrate knowledge and understanding of the key concepts and principles of computer science.
- AO2: Apply knowledge and understanding of key concepts and principles of computer science.
- AO3: Analyse problems in computational terms in order to develop and test programmed solutions and demonstrate an understanding of programming concepts.

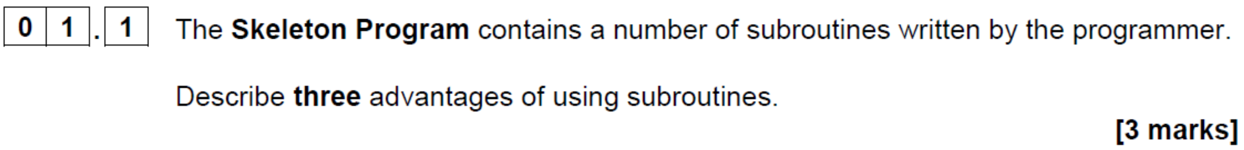
All of the questions on this question paper tested assessment objective AO3.

Example responses

Question number 1

Question number 1 covered various aspects of programming. Some question parts were quite general, such as giving advantages of using subroutines, whilst others related more specifically to code in the Skeleton Program.

Question number part 1.1

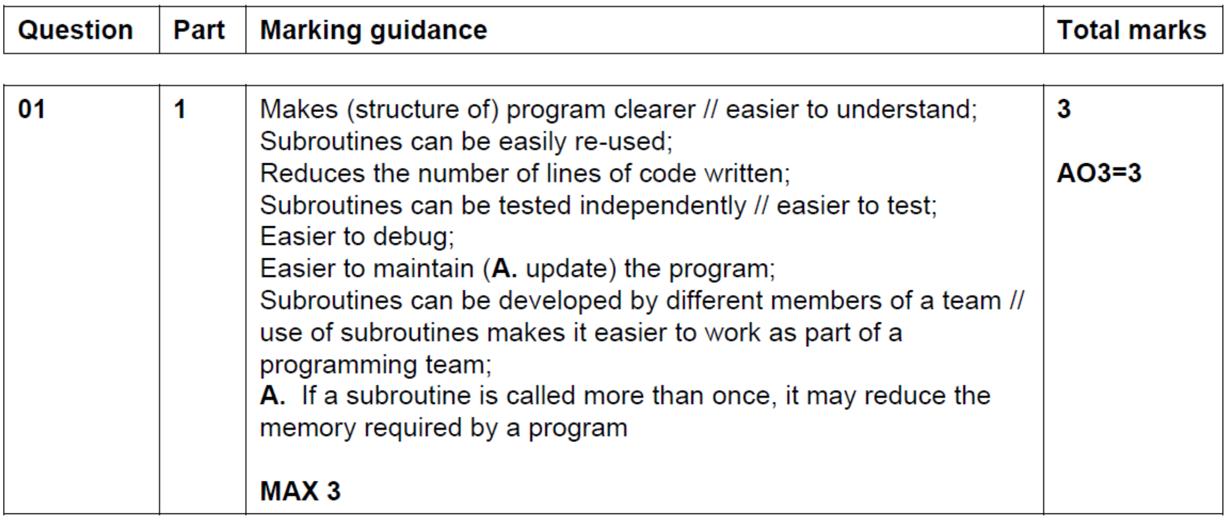


0 1 . 1 The **Skeleton Program** contains a number of subroutines written by the programmer.

Describe **three** advantages of using subroutines.

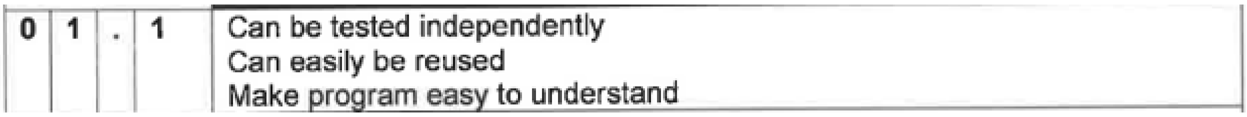
[3 marks]

Mark scheme



Question	Part	Marking guidance	Total marks
01	1	Makes (structure of) program clearer // easier to understand; Subroutines can be easily re-used; Reduces the number of lines of code written; Subroutines can be tested independently // easier to test; Easier to debug; Easier to maintain (**A.** update) the program; Subroutines can be developed by different members of a team // use of subroutines makes it easier to work as part of a programming team; **A.** If a subroutine is called more than once, it may reduce the memory required by a program **MAX 3**	**3** **AO3=3**

Response A

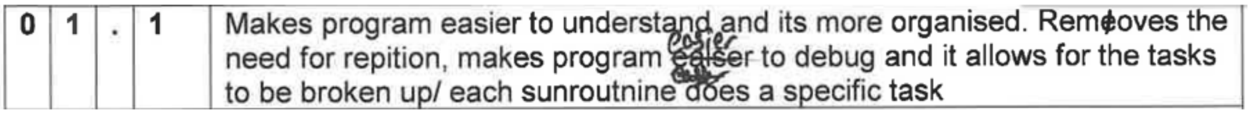


0 1 . 1	Can be tested independently Can easily be reused Make program easy to understand

Commentary

This was a good response, which received **3 marks**. The points were clear and concise.

Response B



0	1	.	1	Makes program easier to understand and its more organised. Rem~~e~~oves the need for repition, makes program ~~ealser~~ easier to debug and it allows for the tasks to be broken up/ each sunroutnine does a specific task

Commentary

This response received **2 marks**. The student has identified that the use of subroutines will make the program easier to understand and also to debug. The point that subroutines “remove the need for repetition” is not enough for a mark. Whilst the use of subroutines will reduce the amount of repeated code in a program, it would not remove the need for repetition.

Question number part 1.2

0 1 . 2 State the name of a variable in the **Skeleton Program** that holds an integer value.

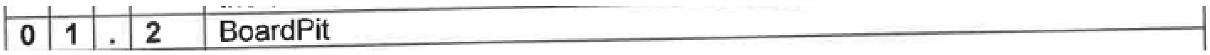
[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks

01	**2**	`Count;`
		`Seeds;`
		`PitCount;`
		`Pit;`
		`DropPit;`
		`LastPit;`
		`Player;`
		`PlayerOneSeeds;`
		`PlayerTwoSeeds;`
		`TotalSeeds;`
		MAX 1
		I. case and spacing
			1
			AO3=1

Response A

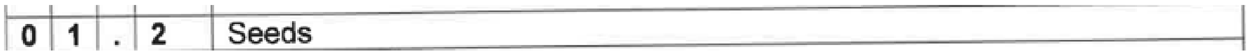


0 1 . 2 BoardPit

Commentary

This response received **0 marks**. `BoardPit` is not the name of a variable in the Skeleton program.

Response B



0 1 . 2 Seeds

Commentary

This response received **1 mark**. The variable named did hold an integer value.

Question number part 1.3

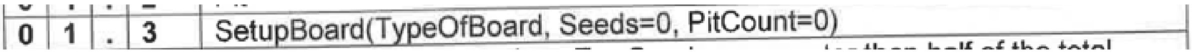
0 1 . 3 State the name of a programmer–written subroutine in the **Skeleton Program** that has **exactly** three parameters.

[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks
01	3	`SetupBoard`; **I.** case and spacing	1 **AO3=1**

Response A

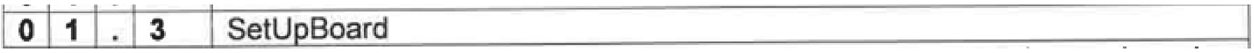


0 1 . 3 SetupBoard(TypeOfBoard, Seeds=0, PitCount=0)

Commentary

This response received **0 marks** because the student has included the list of parameters alongside the subroutine name. When a question asks for the name of a subroutine to be stated, it is important that no other code apart from the subroutine name is given in the response. If more code is given, it is no longer clear that the student knows what the subroutine name is, and so the mark will not be awarded.

Response B



0 1 . 3 SetUpBoard

Commentary

This response received **1 mark**. The student has correctly named a subroutine that had three parameters. There is a minor case error – the “U” should be lowercase, but minor errors like this are ignored when responses are marked.

Question number part 1.4

0 1 . 4 The `GetGameState` subroutine uses a selection statement to determine the value to return.

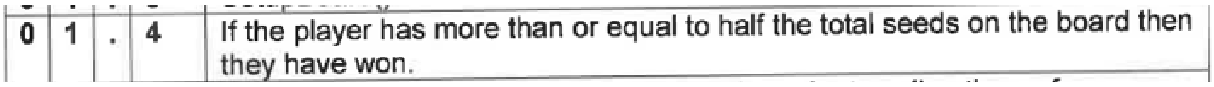
Describe the circumstances under which the string `Won` is returned.

[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks
01	4	(`Won` is returned) when one player has collected more than half of the seeds;	1 AO3=1

Response A

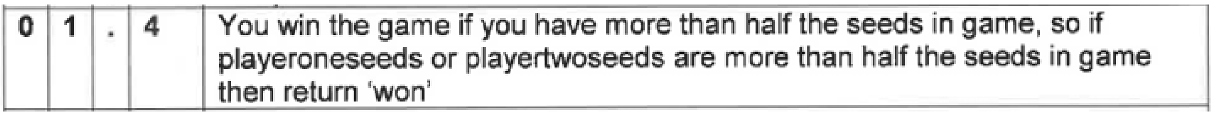


0 1 . 4 If the player has more than or equal to half the total seeds on the board then they have won.

Commentary

This response received **0 marks**. The student has stated “more than or equal to” when the condition in the code was just more than, so the mark could not be awarded.

Response B



0 1 . 4 You win the game if you have more than half the seeds in game, so if playeroneseeds or playertwoseeds are more than half the seeds in game then return ‘won’

Commentary

This response received **1 mark**. The student has clearly described the circumstances under which `Won` is returned.

Question number part 1.5

0 1 . 5 The board is displayed as two rows, but a one-dimensional array has been chosen to represent this instead of a two-dimensional array.

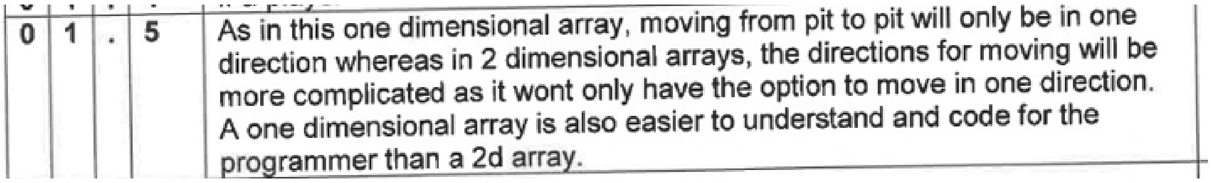
State **two** reasons why using a one-dimensional array instead of a two-dimensional array makes it easier to program, moving from pit to pit.

[2 marks]

Mark scheme

Question	Part	Marking guidance	Total marks
01	5	A two-dimensional array would require the use of an index for row and an index for column (which is harder to program) // A one-dimensional array will only need one index (which is easier to deal with/program); **A.** clearly written answers that talk about movement instead of indexes; Wrapping around from the beginning/end to the end/beginning of the board is easier with a one-dimensional array // moving from one row to another is easier; The game board is a loop/list rather than a table/grid;	**2** **AO3=2**

Response A

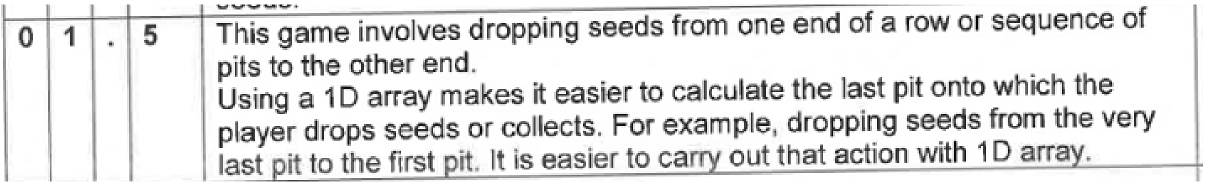


0 1 . 5 As in this one dimensional array, moving from pit to pit will only be in one direction whereas in 2 dimensional arrays, the directions for moving will be more complicated as it wont only have the option to move in one direction.
A one dimensional array is also easier to understand and code for the programmer than a 2d array.

Commentary

This response received **1 mark**. The student has identified that it will be simpler to program as only one direction of movement was needed, which maps to the first mark point on the mark scheme. The response could be improved by linking this to the fact that although the board is displayed as a gird, it is fundamentally a one-dimensional list of pits.

Response B



0 1 . 5 This game involves dropping seeds from one end of a row or sequence of pits to the other end.
Using a 1D array makes it easier to calculate the last pit onto which the player drops seeds or collects. For example, dropping seeds from the very last pit to the first pit. It is easier to carry out that action with 1D array.

Commentary

This response received **1 mark**. The student has identified that it will be easier to move from the last pit to the first pit, which corresponds to the second mark point on the mark scheme, that it would be easier to wrap around from the end to the beginning.

Question number 2

Question Number 2 asked students to explain various aspects of the `DropSeeds` subroutine in the Skeleton Program and to complete a trace table.

Question number part 2.1

0 2 This question is about the `DropSeeds` subroutine.

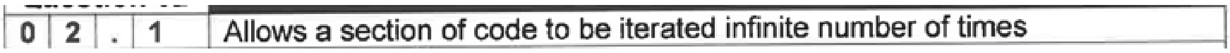
0 2 . 1 Explain what is meant by indefinite iteration.

[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks
02	**1**	(Indefinite iteration is) where the loop is controlled by criteria (which is tested every iteration) // iteration that loops an unknown/variable number of times // iteration that loops while a condition is true/false;	**1** **AO3=1**

Response A

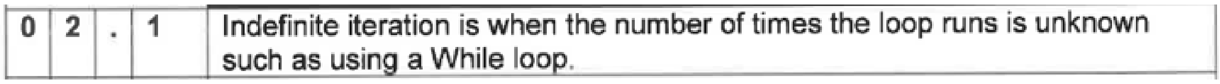


0 2 . 1 Allows a section of code to be iterated infinite number of times

Commentary

This response received **0 marks**. Indefinite iteration could allow some code to iterate an infinite number of times, but the key point that the student needed to make was that the number of times the code iterated would be unknown or variable.

Response B



0 2 . 1 Indefinite iteration is when the number of times the loop runs is unknown such as using a While loop.

Commentary

This response received **1 mark**. The student has correctly identified that the number of times the code would iterate is not known. The student could have improved the explanation by stating that this was because the loop was controlled by a condition and the code in the loop would be repeated for as long as the condition was true, but this was not necessary as there was only one mark available for this question part.

Question number part 2.2

0 2 . 2 The `DropSeeds` subroutine uses indefinite iteration.

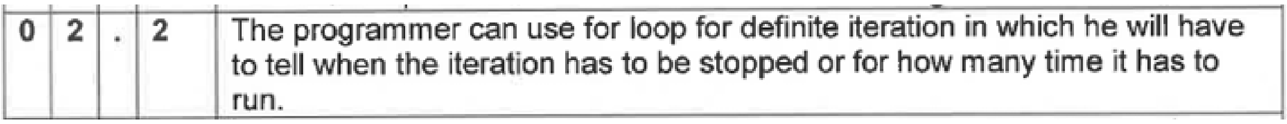
Explain why this could be changed to make use of definite iteration.

[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks
02	2	The loop could be definite iteration as it is known how many times to repeat the loop; We know the number of seeds to drop; The loop does use / can be controlled by a counter; The values of Seeds changes by one each time (so a for loop could be used) **MAX 1**	1 AO3=1

Response A

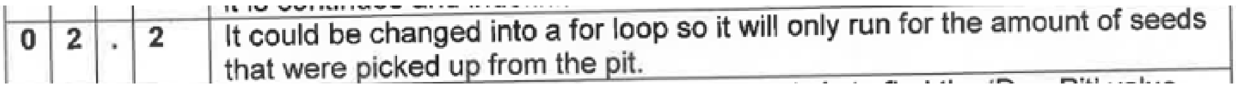


0	2	.	2	The programmer can use for loop for definite iteration in which he will have to tell when the iteration has to be stopped or for how many time it has to run.

Commentary

This response received **0 marks**. The student has attempted to describe what definite iteration is, by stating that the programmer will have to “tell” how many times the code has to run. However, they have not linked this to the context in the Skeleton Program. To achieve a mark, the student needed to explain why this was appropriate to use in the `DropSeeds` subroutine.

Response B



0	2	.	2	It could be changed into a for loop so it will only run for the amount of seeds that were picked up from the pit.

Commentary

This response received **1 mark**. The answer would have been clearer if the student had stated that the number of seeds was known, but this was considered just enough for a mark to be awarded as that statement that “it will only run for the amount of seeds that were picked up” was taken to infer that the number of seeds was known.

Question number part 2.3

0 2 . 3 The MOD operator is written as `%` in C#, `%` in Python, and `Mod` in Visual Basic.

In the subroutine `DropSeeds` the MOD operator is used in two assignment statements.

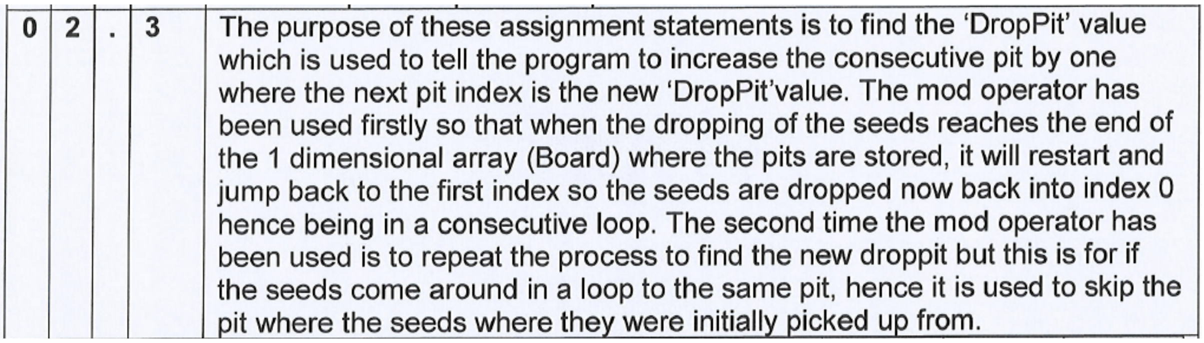
Explain the purpose of these assignment statements and why the MOD operator has been used in them.

[2 marks]

Mark scheme

Question	Part	Marking guidance	Total marks
02	3	The purpose is to move forward one position in the board; (The first) MOD is used so that we move correctly from last index/11 to the next pit // pit 0 // so we wrap around from the end of the board to the beginning; (The second) MOD is used so that we skip the starting pit; **MAX 2**	**2** **AO3=2**

Response A

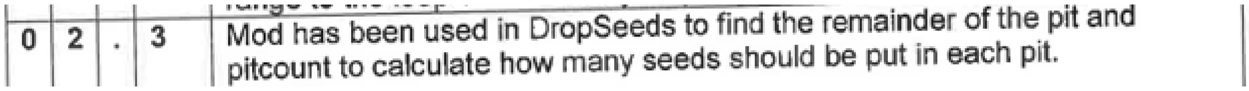


0	2	.	3	The purpose of these assignment statements is to find the 'DropPit' value which is used to tell the program to increase the consecutive pit by one where the next pit index is the new 'DropPit'value. The mod operator has been used firstly so that when the dropping of the seeds reaches the end of the 1 dimensional array (Board) where the pits are stored, it will restart and jump back to the first index so the seeds are dropped now back into index 0 hence being in a consecutive loop. The second time the mod operator has been used is to repeat the process to find the new droppit but this is for if the seeds come around in a loop to the same pit, hence it is used to skip the pit where the seeds where they were initially picked up from.

Commentary

This was a very good response which received **2 marks**. The student correctly identified that the purpose of the code was to move forward one position on the board and that MOD was used to wrap around to the start of the board once the end was reached.

Response B



0	2	.	3	Mod has been used in DropSeeds to find the remainder of the pit and pitcount to calculate how many seeds should be put in each pit.

Commentary

This response received **0 marks**. The student has demonstrated some understanding of the MOD operator by explaining that it will give a remainder. However, their explanation of what it is used for in `DropSeeds` is incorrect. It does not see how many seeds should be put into a pit, so no marks were awarded.

Question number part 2.4

0 2 . 4 Complete the trace table in **Table 1** for the subroutine call `DropSeeds(1)`.

In this game the board has been setup to have only four pits and the current values in `Board` have been filled in for you. The value of the parameter `Pit` has also been filled in for you.

You may not need all the rows in **Table 1**.

Copy the contents of all the unshaded cells in **Table 1** into your Electronic Answer Document.

[6 marks]

Table 1

Board				**Pit**	**Seeds**	**DropPit**	**PitCount**
0	**1**	**2**	**3**				
1	4	1	2	1			

Mark scheme

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>02</td><td>4</td><td>
<table>
<tr><th colspan="4">Board</th><th rowspan="2">Pit</th><th rowspan="2">Seeds</th><th rowspan="2">DropPit</th><th rowspan="2">PitCount</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td>1</td><td>4</td><td>1</td><td>2</td><td>1</td><td></td><td></td><td></td></tr>
<tr><td></td><td>0</td><td></td><td></td><td></td><td>4</td><td>1</td><td>4</td></tr>
<tr><td></td><td></td><td>2</td><td></td><td></td><td>3</td><td>2</td><td></td></tr>
<tr><td></td><td></td><td></td><td>3</td><td></td><td>2</td><td>3</td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td>1</td><td>0</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td><td></td><td>0</td><td>2</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

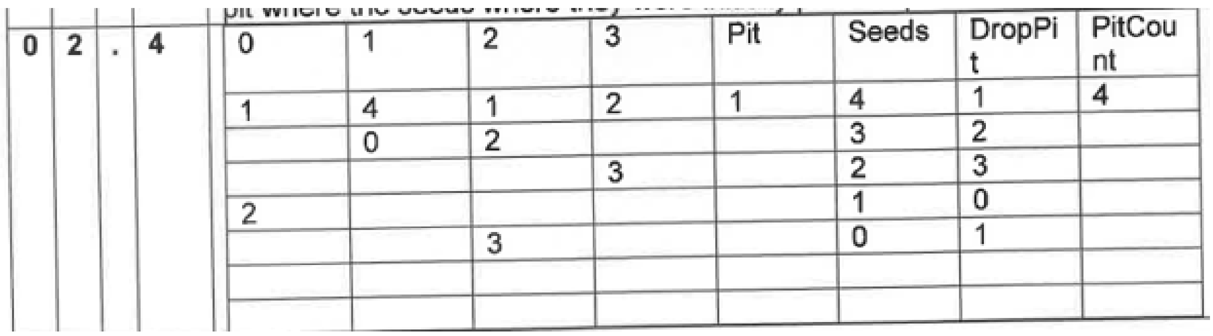
1 mark: Initial values for Seeds, DropPit and PitCount completed correctly
1 mark: Board[1] changes to 0
1 mark: Board[2] changes to 2
1 mark: Seeds column changes to 3 then 2, 1, 0 and has no other values
1 mark: DropPit column changes to 2 then 3, 0, 1, 2 and has no other values
1 mark: Board has correct final values

NOTE: Candidate might use more rows but the value changes should happen in the same order as shown.

MAX 5 if any incorrect / extra values in table
</td><td>6

AO3=6</td></tr>
</table>

Response A



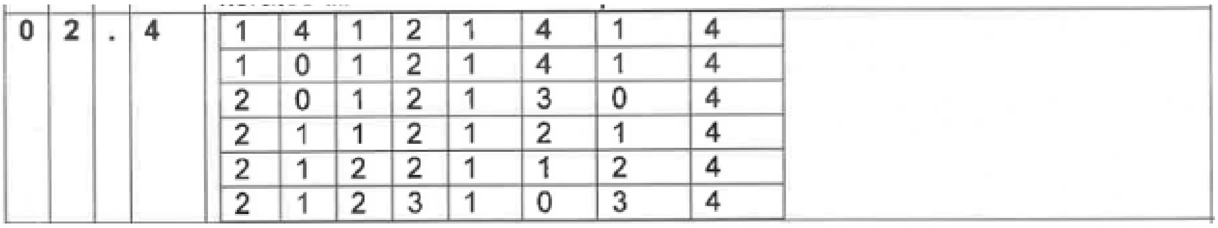
0 2 . 4

0	1	2	3	Pit	Seeds	DropPit	PitCount
1	4	1	2	1	4	1	4
	0	2			3	2	
			3		2	3	
2					1	0	
		3			0	1	

Commentary

This was a good response which received **5 marks** of the available 6 marks. The student has just missed writing the final value 2 at the bottom of the `DropPit` column so did not receive the final mark. Missing off the last value in a column is a relatively common error, often caused by students stopping completing the table when they can see that the algorithm has achieved its purpose, instead of when the end of the code has been reached.

Response B



0 2 . 4

1	4	1	2	1	4	1	4
1	0	1	2	1	4	1	4
2	0	1	2	1	3	0	4
2	1	1	2	1	2	1	4
2	1	2	2	1	1	2	4
2	1	2	3	1	0	3	4

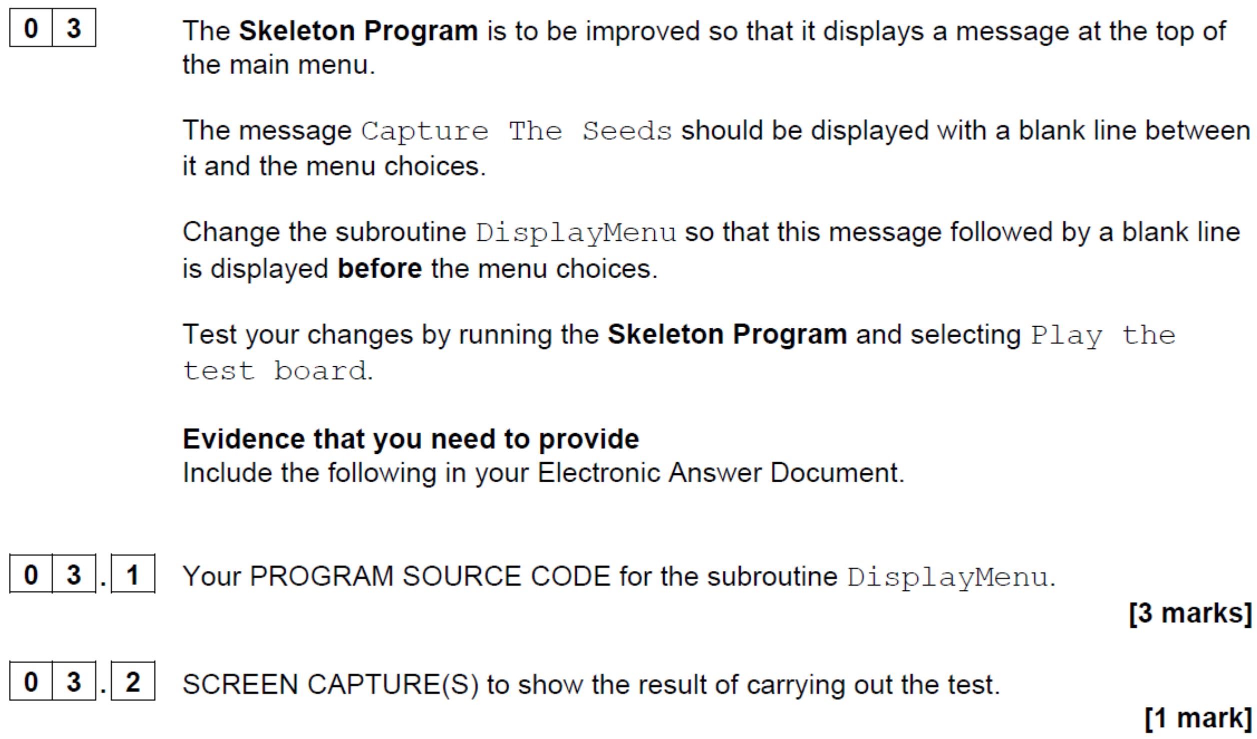
Commentary

This response received **4 marks**, for the first four mark points on the mark scheme. This student has written out the values of some variables more frequently than necessary, but this has not affected their mark. It is only necessary to write the value of a variable when it changes.

Question number 3

For Question Number 3, students had to modify the `DisplayMenu` subroutine in the Skeleton Program so that it displayed an additional message.

Question number part 3.1 and 3.2



0 3 The **Skeleton Program** is to be improved so that it displays a message at the top of the main menu.

The message `Capture The Seeds` should be displayed with a blank line between it and the menu choices.

Change the subroutine `DisplayMenu` so that this message followed by a blank line is displayed **before** the menu choices.

Test your changes by running the **Skeleton Program** and selecting `Play the test board`.

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

0 3 . 1 Your PROGRAM SOURCE CODE for the subroutine `DisplayMenu`.

[3 marks]

0 3 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks

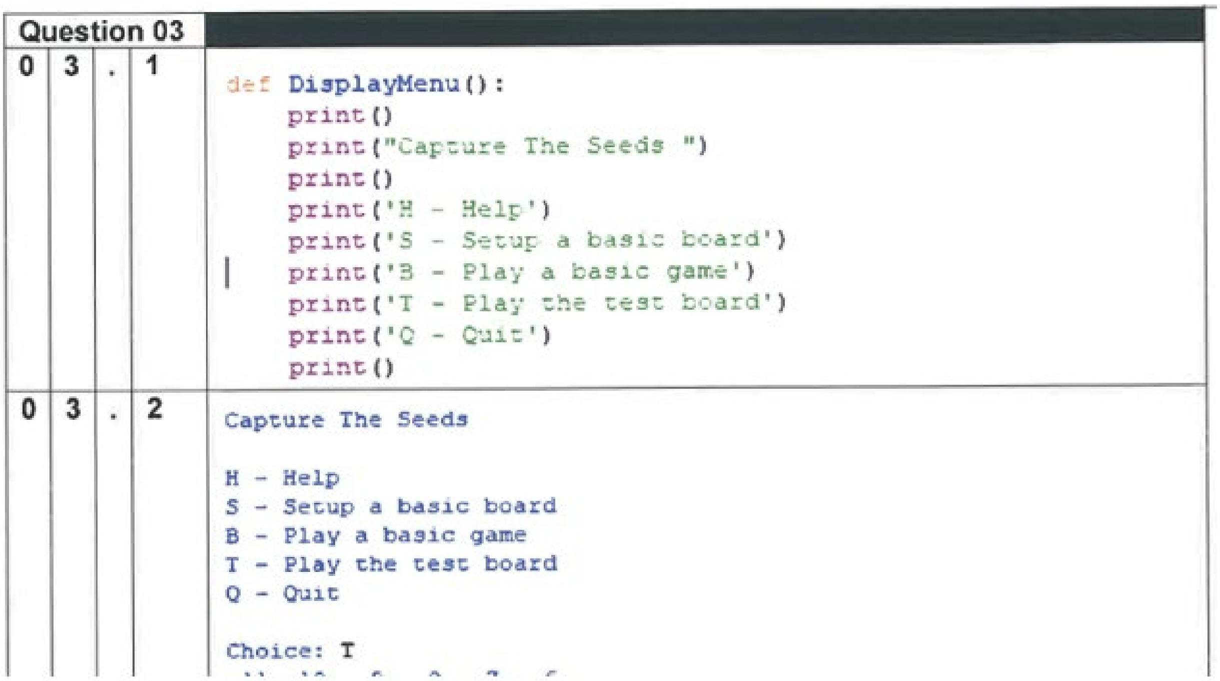
03	1	**1 mark:** Program displays correct message.
		A. minor typos and incorrect case.
		I. full stop at end of message
		
		1 mark: Statements are at a position so that message is displayed whenever the menu is displayed and before the menu options;
		
		1 mark: Blank line is printed after the message is displayed;
		
		MAX 2 if any errors
		
		Python
		`def Menu():`
		`    print()`
		`    print('Capture The Seeds')`
		`    print()`
		`    print('H - Help')`
		`    print('S - Setup a basic board')`
		
		VB
		`Sub DisplayMenu()`
		`    Console.WriteLine()`
		`    Console.WriteLine("Capture The Seeds")`
		`    Console.WriteLine()`
		`    Console.WriteLine("H - Help")`
		`    Console.WriteLine("S - Setup a basic board")`
		`    Console.WriteLine("B - Play a basic game")`
		`    Console.WriteLine("T - Play the test board")`
		`    Console.WriteLine("Q - Quit")`
		`    Console.WriteLine()`
		`End Sub`
		
		C#
		`private static void DisplayMenu()`
		`{`
		`    Console.WriteLine();`
		`    Console.WriteLine("Capture The Seeds");`
		`    Console.WriteLine();`
		`    Console.WriteLine("H - Help");`
		`    Console.WriteLine("S - Setup a basic board");`
			3
			
			AO3=3

```
    Console.WriteLine("B - Play a basic
game");
    Console.WriteLine("T - Play the test
board");
    Console.WriteLine("Q - Quit");
    Console.WriteLine();
}
```

Question	Part	Marking guidance	Total marks
03	2	Screen capture(s) must match code from Q3.1 **1 mark:** Correct message displayed above menu.	1 **AO3=1**

```
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from?
```

Response A



Question 03

0 3 . 1

```
def DisplayMenu():
    print()
    print("Capture The Seeds ")
    print()
    print('H - Help')
    print('S - Setup a basic board')
    print('B - Play a basic game')
    print('T - Play the test board')
    print('Q - Quit')
    print()
```

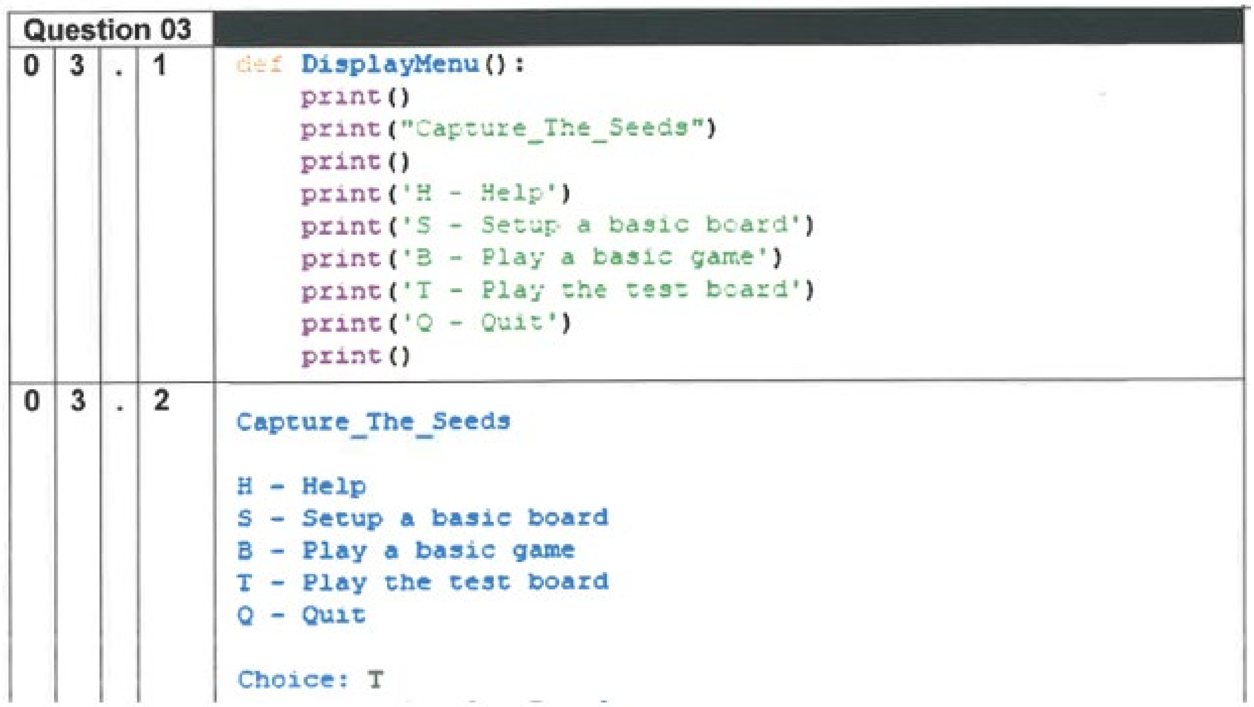
0 3 . 2

```
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
```

Commentary

This response received **3 marks** for part **3.1** and **1 mark** for part **3.2**. The student has displayed the correct message in the correct place, with a blank line after it and has shown this in their test evidence, so full marks were awarded.

Response B



Question 03

0 3 . 1

```
def DisplayMenu():
    print()
    print("Capture_The_Seeds")
    print()
    print('H - Help')
    print('S - Setup a basic board')
    print('B - Play a basic game')
    print('T - Play the test board')
    print('Q - Quit')
    print()
```

0 3 . 2

```

Capture_The_Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
```

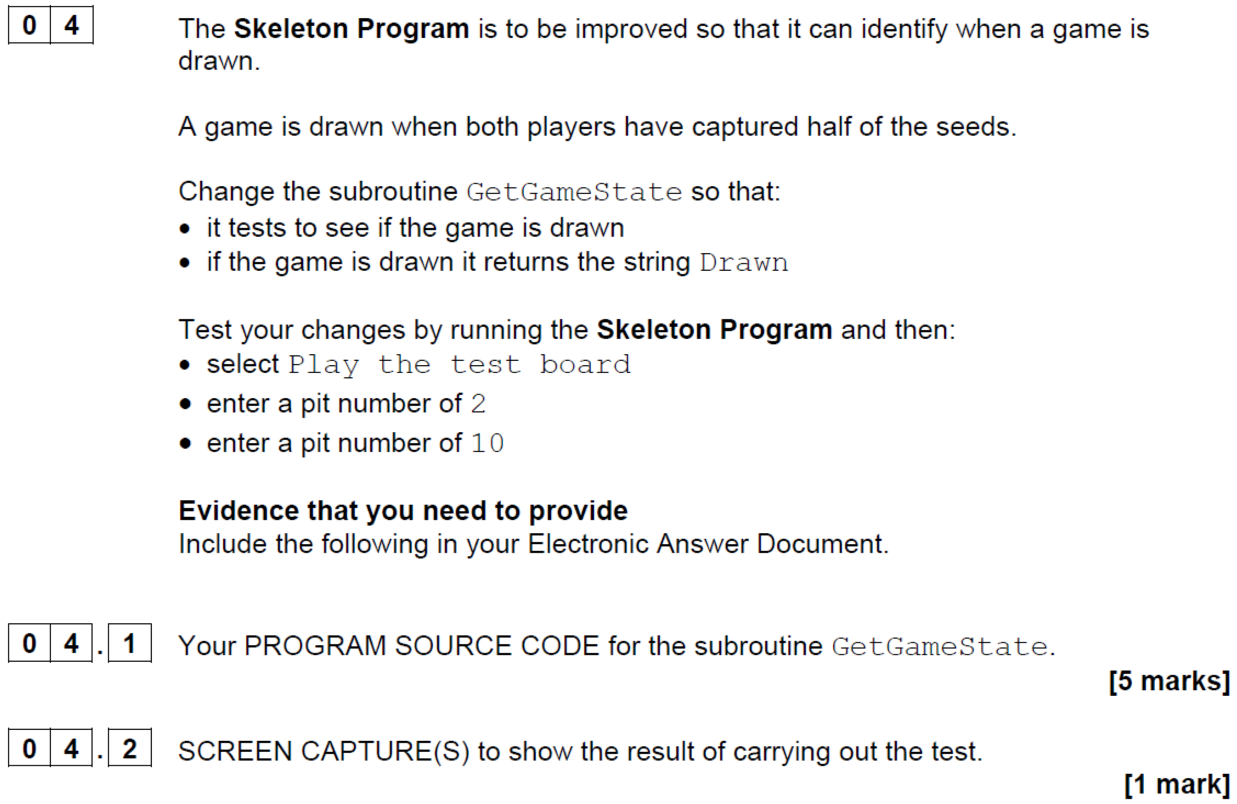
Commentary

This response received **3 marks** for part **3.1** and **1 mark** for part **3.2**. The message displayed was not quite correct (the spaces were replaced by underscores) but minor mistakes in the message were accepted when this question was marked.

Question number 4

Question number 4 asked students to change the `GetGameState` subroutine in the Skeleton Program so that it would detect a drawn game.

Question number part 4.1 and 4.2



0 4 The **Skeleton Program** is to be improved so that it can identify when a game is drawn.

A game is drawn when both players have captured half of the seeds.

Change the subroutine `GetGameState` so that:

- it tests to see if the game is drawn
- if the game is drawn it returns the string `Drawn`

Test your changes by running the **Skeleton Program** and then:

- select `Play the test board`
- enter a pit number of `2`
- enter a pit number of `10`

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

0 4 . 1 Your PROGRAM SOURCE CODE for the subroutine `GetGameState`.

[5 marks]

0 4 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Mark scheme

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>04</td><td>1</td><td>

1 mark: Check `State[1]` / `PlayerOneSeeds` has half of the seeds
1 mark: Check `State[2]` / `PlayerTwoSeeds` has half of the seeds
1 mark: Correctly combine the two checks.

or

1 mark: check `State[1]` / `PlayerOneSeeds` / `State[2]` / `PlayerTwoSeeds` has half of the seeds
1 mark: Check `PlayerOneSeeds` and `PlayerTwoSeeds` are the same // Check sum of `PlayerOneSeeds` and `PlayerTwoSeeds` is equal to `TotalSeeds`
1 mark: Correctly combine the two checks.

or

1 mark: attempt to calculate the sum of the values in the `Board` array/list
1 mark: correctly calculates the sum of the values in the `Board` array/list
1 mark: compares calculated total with 0

and

1 mark: Return the string `Drawn`
1 mark: Only return the string `Drawn` under correct circumstances

MAX 4 if any errors

Python

```
def GetGameState(Player):
    global State
    TotalSeeds = State[0]
    PlayerOneSeeds = State[1]
    PlayerTwoSeeds = State[2]
    if  PlayerOneSeeds >= TotalSeeds // 2 + 1 or
PlayerTwoSeeds >= TotalSeeds // 2 + 1:
        return 'Won'
    elif PlayerOneSeeds == TotalSeeds // 2 and
PlayerTwoSeeds == TotalSeeds // 2:
        return 'Drawn'
    else:
        return 'Play'
```

</td><td>5

AO3=5</td></tr>
</table>

Alternative answer

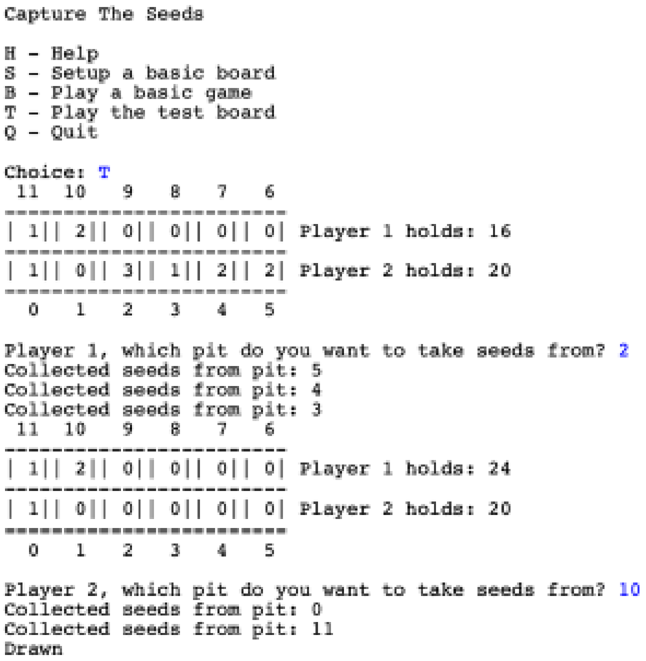
```
def GetGameState(Player):
    global State
    TotalSeeds = State[0]
    PlayerOneSeeds = State[1]
    PlayerTwoSeeds = State[2]
    if  PlayerOneSeeds >= TotalSeeds // 2 + 1 or
PlayerTwoSeeds >= TotalSeeds // 2 + 1:
        return 'Won'
    elif PlayerOneSeeds + PlayerTwoSeeds ==
TotalSeeds:
        return 'Drawn'
    else:
        return 'Play'
```

VB

```
Function GetGameState(Player As Integer) As
String
    Dim TotalSeeds, PlayerOneSeeds,
PlayerTwoSeeds As Integer
    TotalSeeds = State(0)
    PlayerOneSeeds = State(1)
    PlayerTwoSeeds = State(2)
    If PlayerOneSeeds >= TotalSeeds \ 2 + 1 Or
PlayerTwoSeeds >= TotalSeeds \ 2 + 1 Then
        Return "Won"
    ElseIf PlayerOneSeeds = TotalSeeds \ 2 And
PlayerTwoSeeds = TotalSeeds \ 2 Then
        Return "Drawn"
    Else
        Return "Play"
    End If
End Function
```

C#

```
private static string GetGameState(int Player)
{
    int TotalSeeds, PlayerOneSeeds,
PlayerTwoSeeds;
    TotalSeeds = State[0];
    PlayerOneSeeds = State[1];
    PlayerTwoSeeds = State[2];
    if (PlayerOneSeeds >= TotalSeeds / 2 + 1 ||
PlayerTwoSeeds >= TotalSeeds / 2 + 1)
    {
        return "Won";
    }
    else if (PlayerOnesSeeds == TotalSeeds / 2
&& PlayerTwoSeeds == TotalSeeds / 2)
    {
        return "Drawn";
    }
        return "Play";
}
```

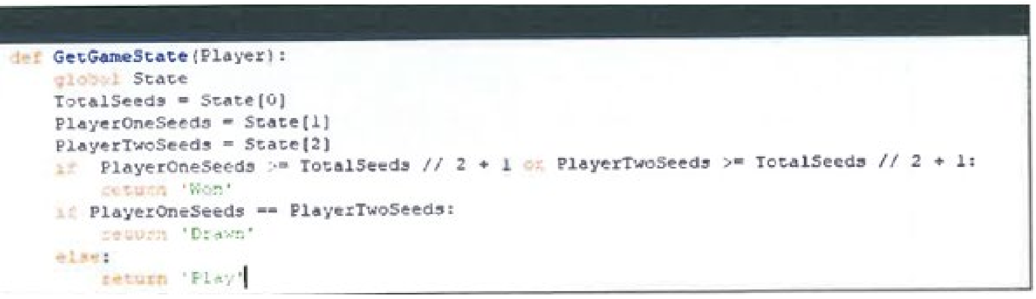
Question	Part	Marking guidance	Total marks
04	2	Screen capture(s) must match code from Q4.1 **1 mark:** Screen capture shows the test game being played with the requested moves ending in a `Drawn` message. 	1 AO3=1

```
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
 11  10   9   8   7   6
========================
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
========================
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
========================
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 2
Collected seeds from pit: 5
Collected seeds from pit: 4
Collected seeds from pit: 3
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 24
------------------------
| 1|| 0|| 0|| 0|| 0|| 0| Player 2 holds: 20
========================
  0   1   2   3   4   5

Player 2, which pit do you want to take seeds from? 10
Collected seeds from pit: 0
Collected seeds from pit: 11
Drawn
```

Response A

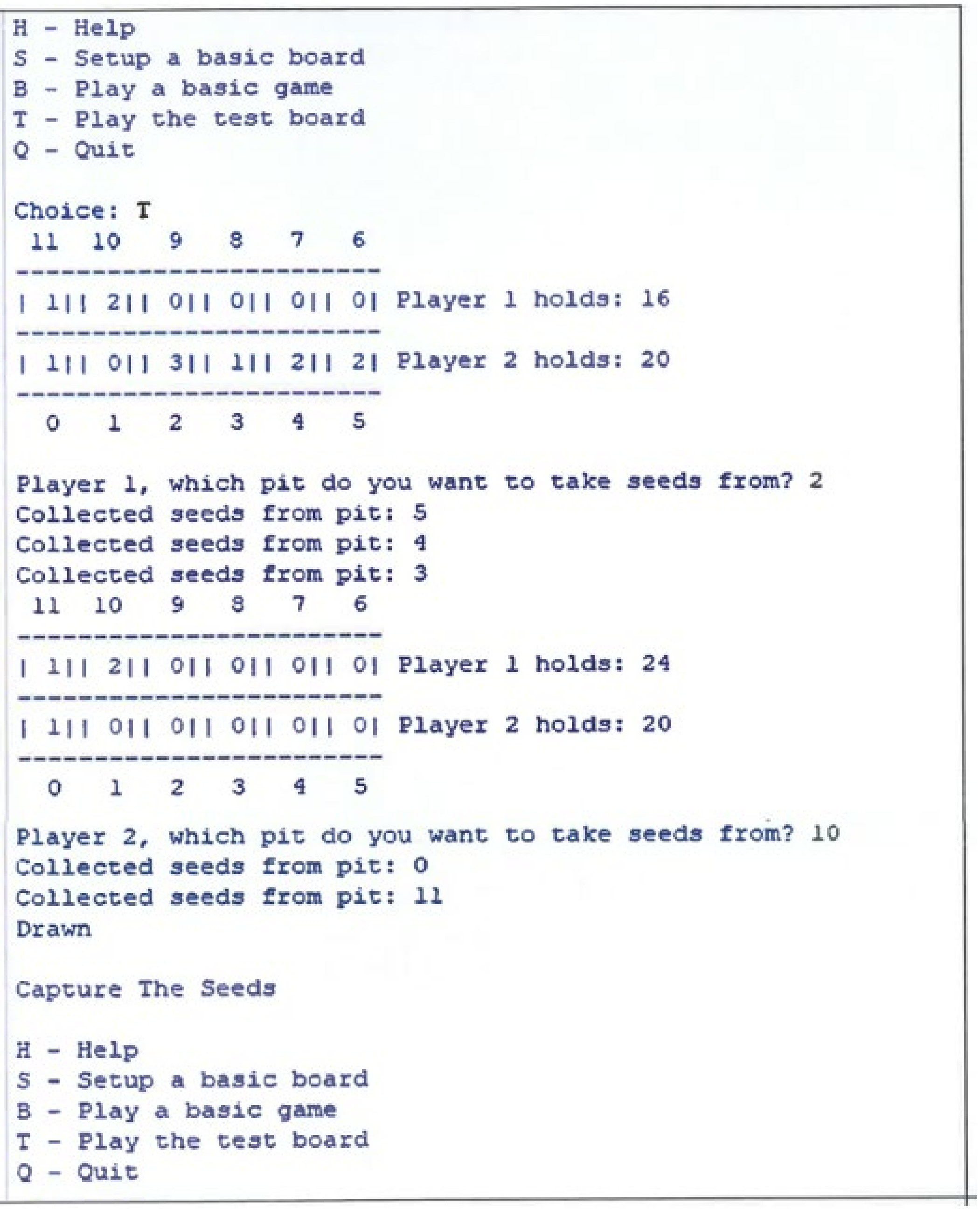
Question 04

0 4 . 1



```
def GetGameState(Player):
    global State
    TotalSeeds = State[0]
    PlayerOneSeeds = State[1]
    PlayerTwoSeeds = State[2]
    if  PlayerOneSeeds >= TotalSeeds // 2 + 1 or PlayerTwoSeeds >= TotalSeeds // 2 + 1:
        return 'Won'
    if PlayerOneSeeds == PlayerTwoSeeds:
        return 'Drawn'
    else:
        return 'Play'
```

0 4 . 2



```
H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 2
Collected seeds from pit: 5
Collected seeds from pit: 4
Collected seeds from pit: 3
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 24
------------------------
| 1|| 0|| 0|| 0|| 0|| 0| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 2, which pit do you want to take seeds from? 10
Collected seeds from pit: 0
Collected seeds from pit: 11
Drawn

Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit
```

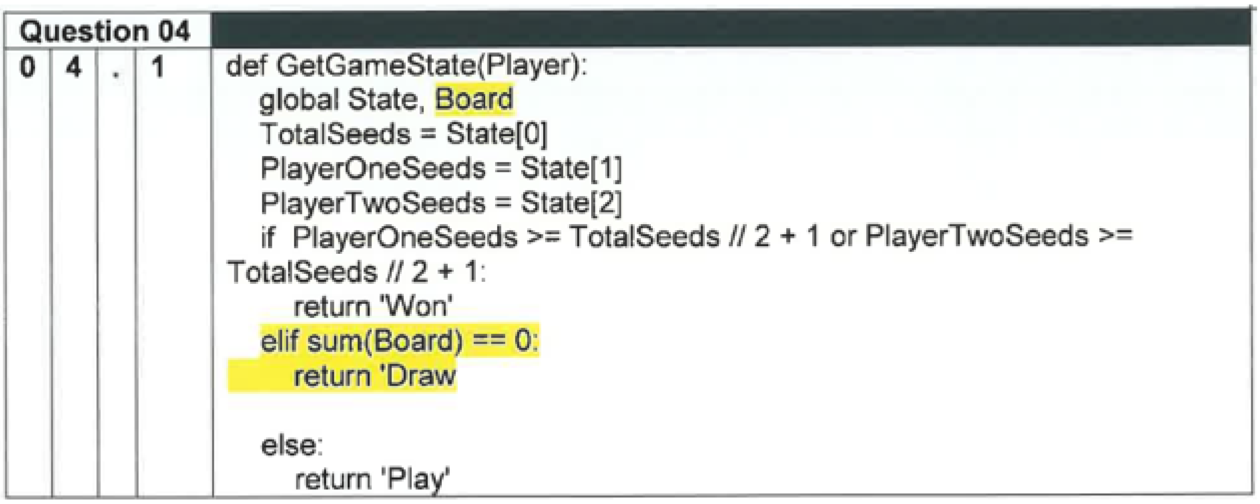
Commentary

This response received **2 marks** for part **4.1** and **1 mark** for part **4.2**.

For part 4.1, the student has checked if the number of seeds each player has is equal to determine if there is a draw, instead of checking that each player has half of the seeds being used in the game. Therefore, none of mark points one to three on the mark scheme can be awarded. However, the subroutine can return the string Drawn, so mark point four can be awarded. The benefit of the doubt has been applied and mark point five has also been awarded as Drawn would only be returned if the student’s criteria for a draw are met. The fact that these criteria are incorrect has not been penalised as the student has already missed three marks because the criteria were incorrect.

The mark for part 4.2 was awarded, as when there was a draw using the test data the screenshots show that “Drawn” was displayed. This mark was awarded despite the fact that “Drawn” would also be displayed in some incorrect circumstances.

Response B

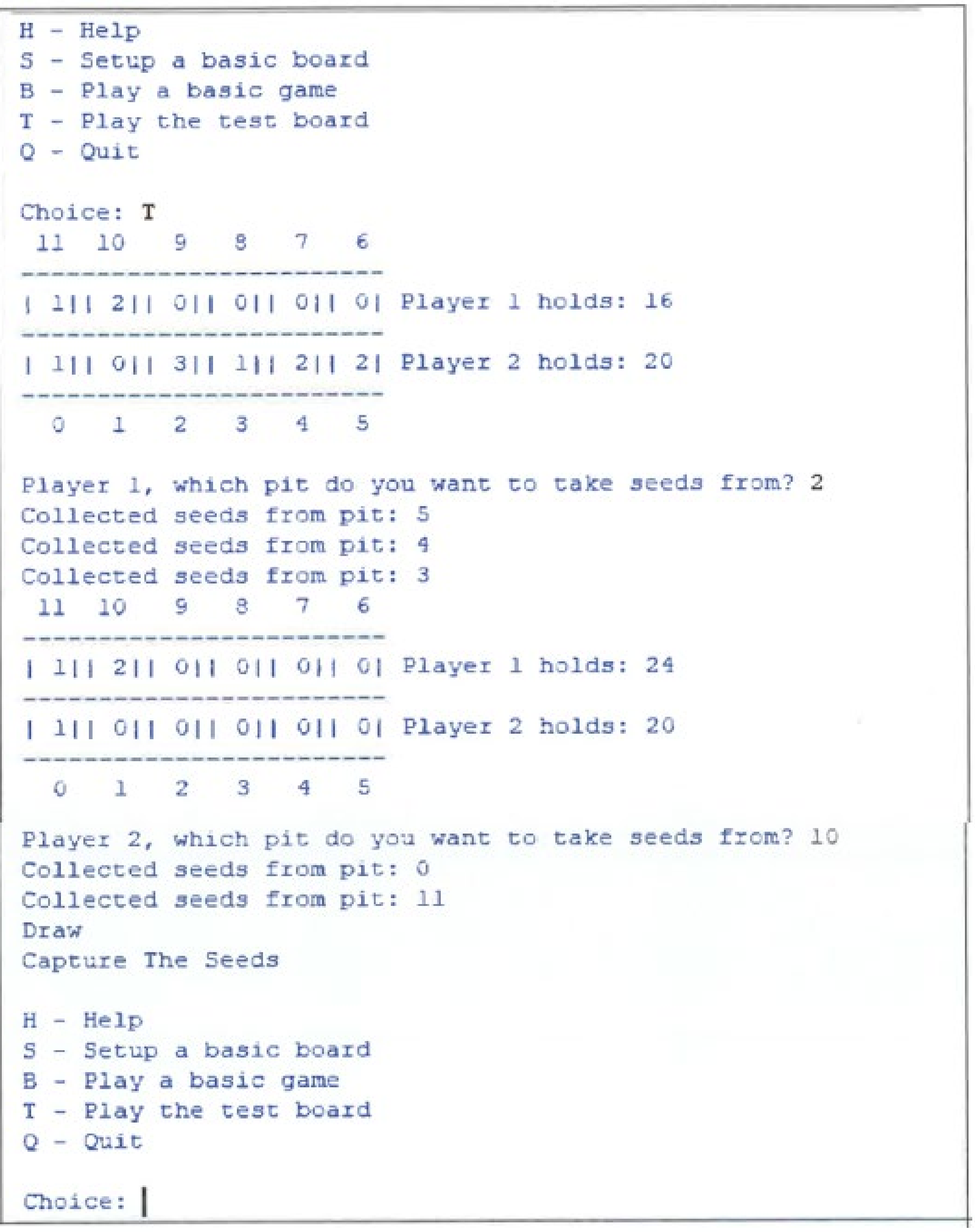


Question 04

0 4 . 1

```
def GetGameState(Player):
    global State, Board
    TotalSeeds = State[0]
    PlayerOneSeeds = State[1]
    PlayerTwoSeeds = State[2]
    if  PlayerOneSeeds >= TotalSeeds // 2 + 1 or PlayerTwoSeeds >=
TotalSeeds // 2 + 1:
        return 'Won'
    elif sum(Board) == 0:
        return 'Draw

    else:
        return 'Play'
```

04.2



```
H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 2
Collected seeds from pit: 5
Collected seeds from pit: 4
Collected seeds from pit: 3
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 24
------------------------
| 1|| 0|| 0|| 0|| 0|| 0| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 2, which pit do you want to take seeds from? 10
Collected seeds from pit: 0
Collected seeds from pit: 11
Draw
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: |
```

Commentary

This response received **5 marks** for part **4.1** and **0 marks** for part **4.2**.

For part 4.1, the student produced an infrequently seen response in which they checked if there was a draw by testing if there were no seeds left on the board, after already checking that no player had more than half of the seeds. If the board had no seeds on it and no player had more than half of the seeds then it follows that each player must have half of the seeds in the game and so the game is a draw. Therefore, despite the student solving the problem in a different way to the examples shown on the mark scheme, all 5 marks were awarded. We will always award marks for responses that work, regardless of whether or not they match the mark scheme as there are often many ways that programming problems can be solved. The code is missing a closing `'` symbol after `Draw`, but this omission was not considered sufficient to prevent the awarding of all of the marks.

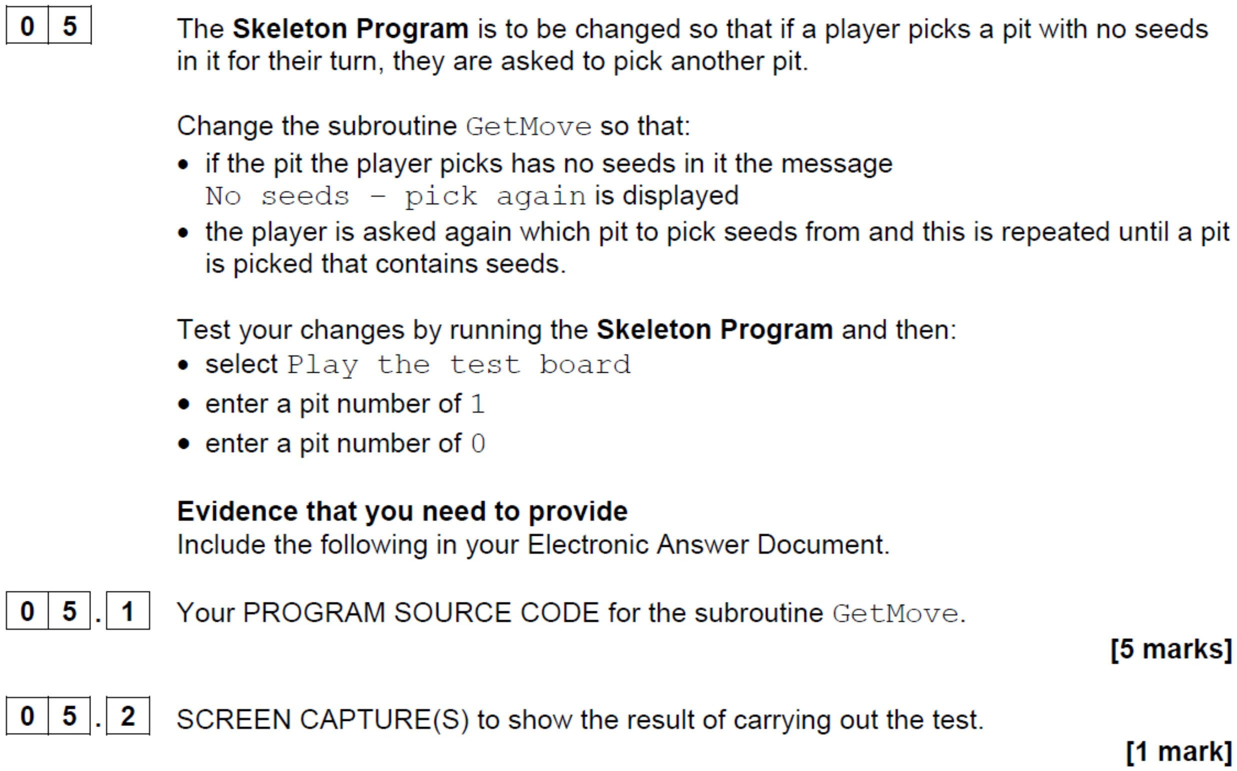
Marks were not awarded for question part 4.2 as the output shown could not have been produced by the student's code owing to the missing `'` symbol. Marks are only awarded for test evidence that would be produced by the student's code.

Students should be careful to copy the full contents of their finished code into the EAD to ensure that they do not accidentally lose some marks.

Question number 5

Question number 5 asked students to change the `GetMove` subroutine so that if a player picked an empty pit to use during their turn they would be made to pick a different pit.

Question number part 5.1 and 5.2



0 5 The **Skeleton Program** is to be changed so that if a player picks a pit with no seeds in it for their turn, they are asked to pick another pit.

Change the subroutine `GetMove` so that:

- if the pit the player picks has no seeds in it the message `No seeds – pick again` is displayed
- the player is asked again which pit to pick seeds from and this is repeated until a pit is picked that contains seeds.

Test your changes by running the **Skeleton Program** and then:

- select `Play the test board`
- enter a pit number of `1`
- enter a pit number of `0`

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

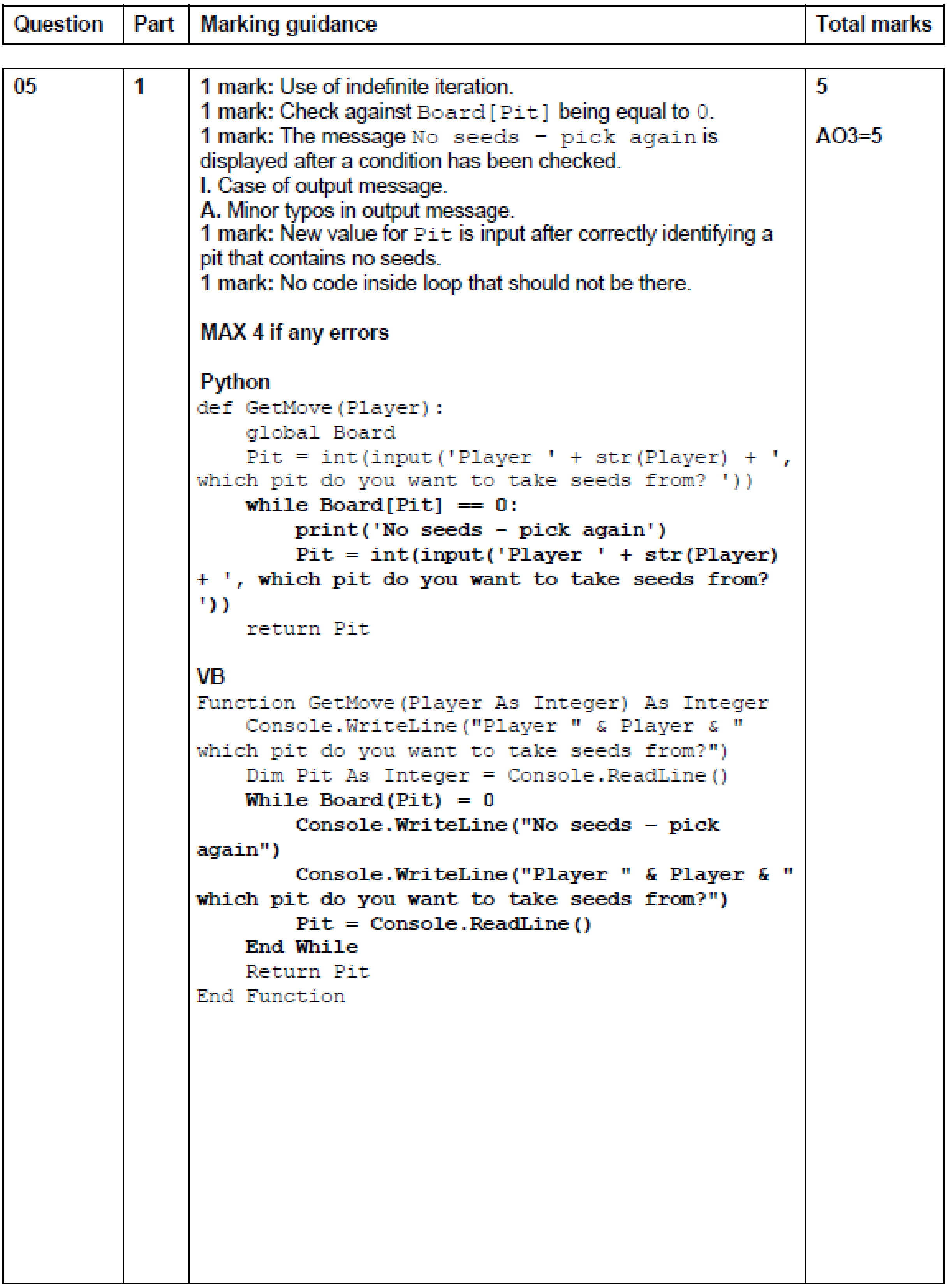
0 5 . 1 Your PROGRAM SOURCE CODE for the subroutine `GetMove`.

[5 marks]

0 5 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Mark scheme

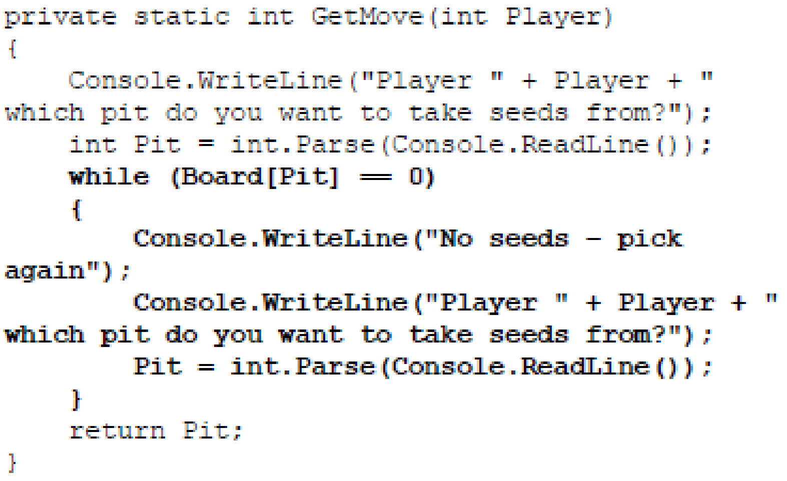


<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>05</td><td>1</td><td>
<b>1 mark:</b> Use of indefinite iteration.
<b>1 mark:</b> Check against <code>Board[Pit]</code> being equal to <code>0</code>.
<b>1 mark:</b> The message <code>No seeds - pick again</code> is displayed after a condition has been checked.
<b>I.</b> Case of output message.
<b>A.</b> Minor typos in output message.
<b>1 mark:</b> New value for <code>Pit</code> is input after correctly identifying a pit that contains no seeds.
<b>1 mark:</b> No code inside loop that should not be there.

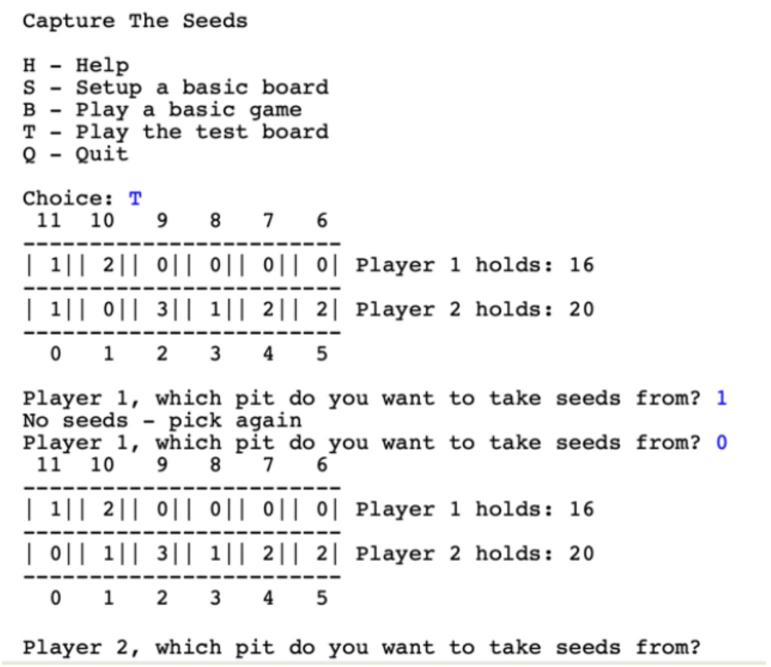
<b>MAX 4 if any errors</b>

<b>Python</b>
<pre>
def GetMove(Player):
    global Board
    Pit = int(input('Player ' + str(Player) + ', which pit do you want to take seeds from? '))
    while Board[Pit] == 0:
        print('No seeds - pick again')
        Pit = int(input('Player ' + str(Player) + ', which pit do you want to take seeds from? '))
    return Pit
</pre>
<b>VB</b>
<pre>
Function GetMove(Player As Integer) As Integer
    Console.WriteLine("Player " & Player & " which pit do you want to take seeds from?")
    Dim Pit As Integer = Console.ReadLine()
    While Board(Pit) = 0
        Console.WriteLine("No seeds - pick again")
        Console.WriteLine("Player " & Player & " which pit do you want to take seeds from?")
        Pit = Console.ReadLine()
    End While
    Return Pit
End Function
</pre>
</td><td>5

AO3=5</td></tr>
</table>

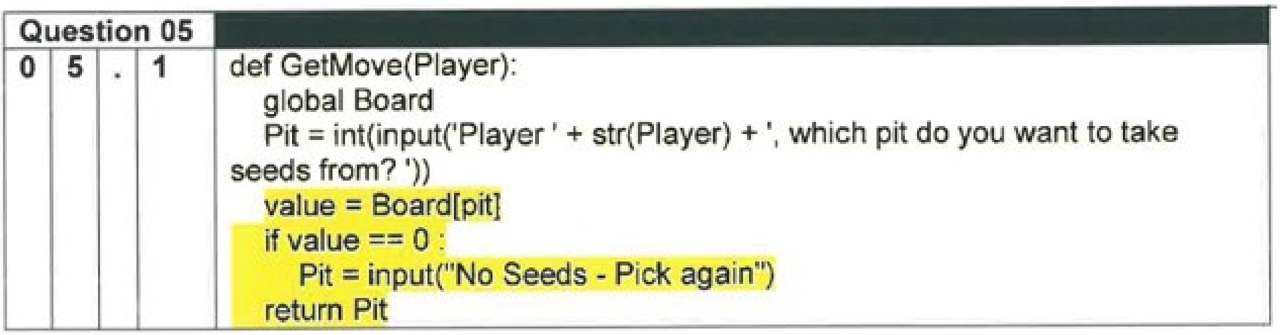
		C# 	

```
private static int GetMove(int Player)
{
    Console.WriteLine("Player " + Player + "
which pit do you want to take seeds from?");
    int Pit = int.Parse(Console.ReadLine());
    while (Board[Pit] == 0)
    {
        Console.WriteLine("No seeds - pick
again");
        Console.WriteLine("Player " + Player + "
which pit do you want to take seeds from?");
        Pit = int.Parse(Console.ReadLine());
    }
    return Pit;
}
```

Question	Part	Marking guidance	Total marks
05	2	Screen capture(s) must match code from Q5.1 **1 mark:** Correct messages shown 	1 **AO3=1**

```
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
   0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 1
No seeds - pick again
Player 1, which pit do you want to take seeds from? 0
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 0|| 1|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
   0   1   2   3   4   5

Player 2, which pit do you want to take seeds from?
```

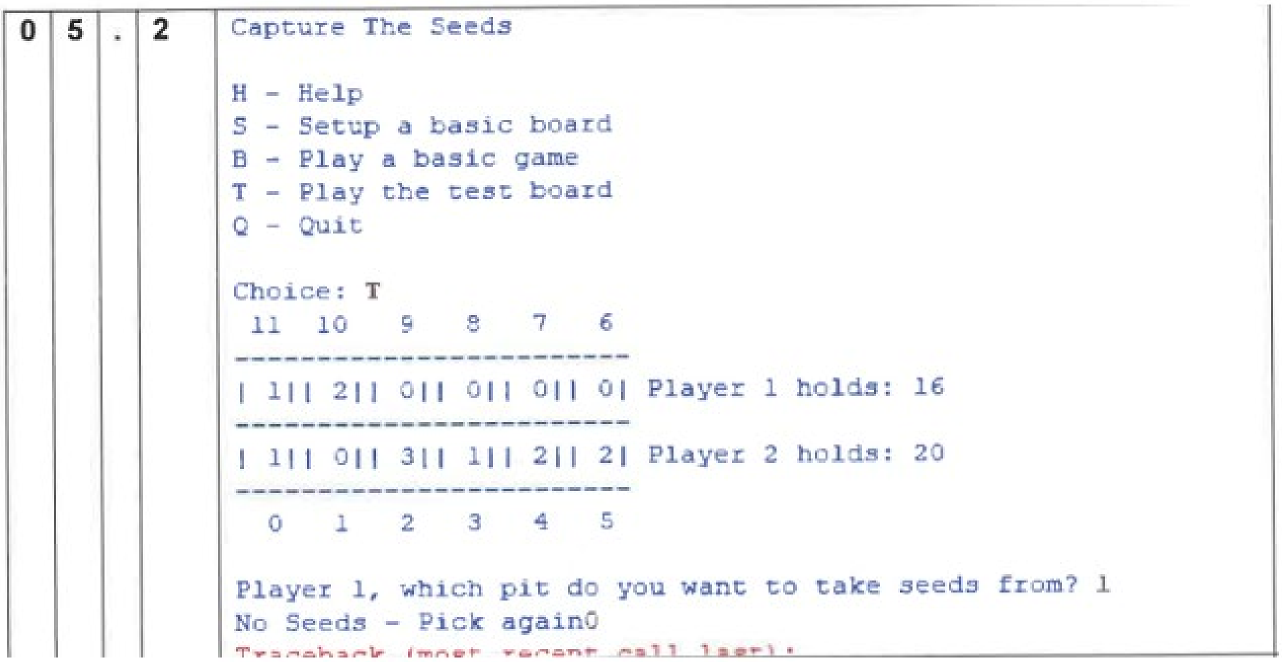
Response A



Question 05

0 5 . 1

```
def GetMove(Player):
    global Board
    Pit = int(input('Player ' + str(Player) + ', which pit do you want to take
seeds from? '))
    value = Board[pit]
    if value == 0 :
        Pit = input("No Seeds - Pick again")
    return Pit
```



0 5 . 2

```
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 1
No Seeds - Pick again0
Traceback (most recent call last):
```

Commentary

This response received **2 marks** for part **5.1** and **0 marks** for part **5.2**.

For part 5.1, no iteration takes place so mark point one from the mark scheme cannot be awarded. The question asked that the process was “repeated until a pit is picked that contains seeds”, so the student should have realised that a loop was required.

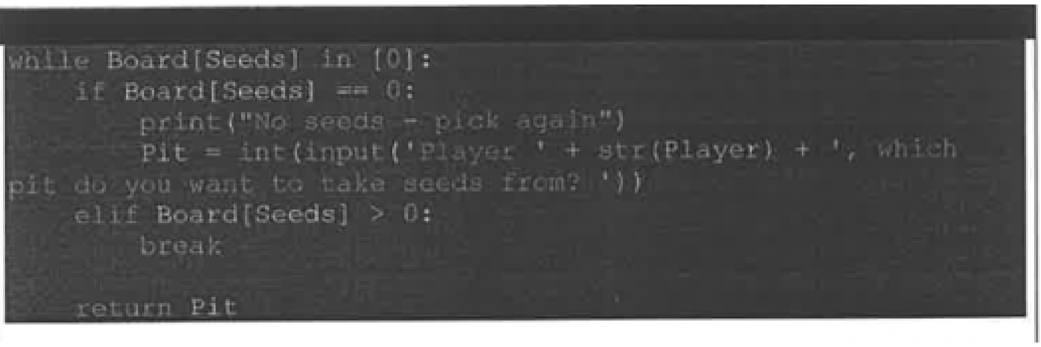
The student does check if `Board[Pit]` is equal to zero; an unnecessary variable `value` is used as part of this process, but the same outcome is achieved, so mark point two from the mark scheme can be awarded. The correct error message is displayed if `value` is equal to 0, so mark point three can also be awarded. Mark point four cannot be awarded as the value for `Pit` is not successfully input because the program does not convert the input into an integer. Mark point five also cannot be awarded as there is no loop.

For part 5.2, no marks can be awarded as the error message is not displayed under the correct circumstances – in fact, it can be seen that the code does not execute.

Response B

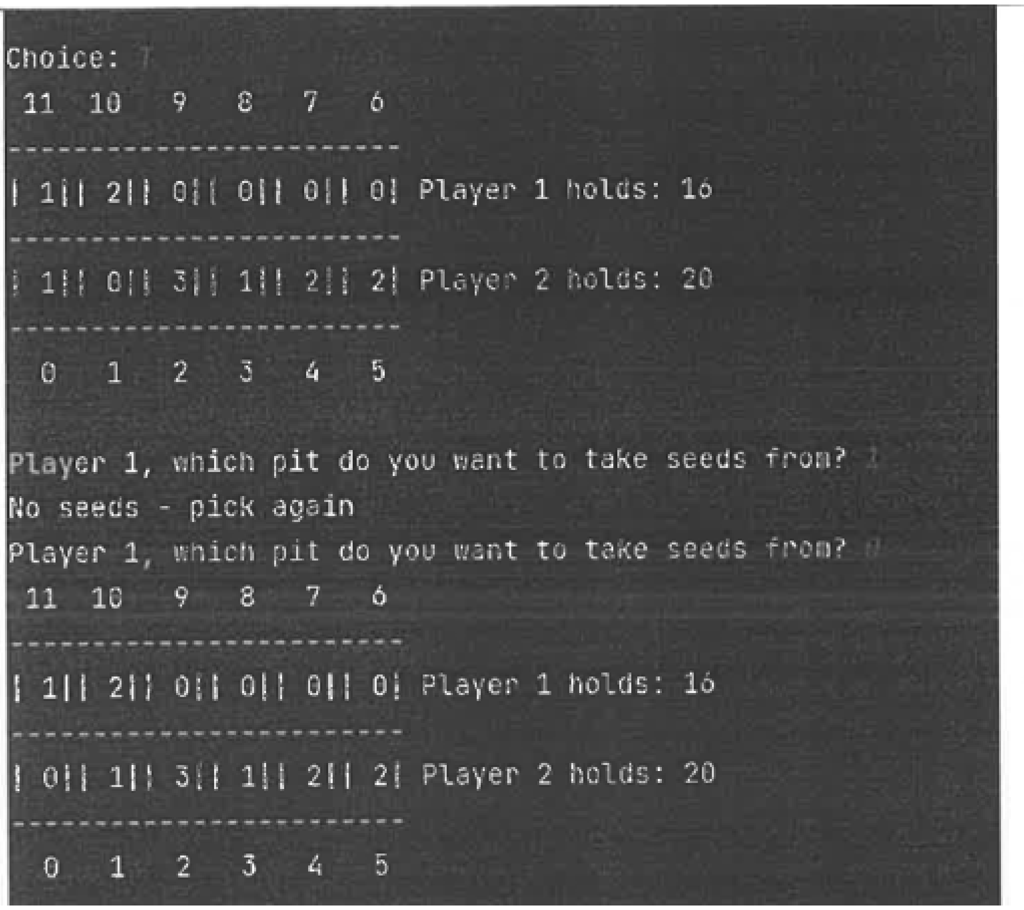
Question 05

0 5 . 1



```
while Board[Seeds] in [0]:
    if Board[Seeds] == 0:
        print("No seeds - pick again")
        Pit = int(input('Player ' + str(Player) + ', which
pit do you want to take seeds from? '))
    elif Board[Seeds] > 0:
        break

    return Pit
```

0 5 . 2



```
Choice: 7
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 1
No seeds - pick again
Player 1, which pit do you want to take seeds from? 0
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 0|| 1|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
  0   1   2   3   4   5
```

Commentary

This was a very good response that received **5 marks** for part **5.1** and **1 mark** for part **5.2**.

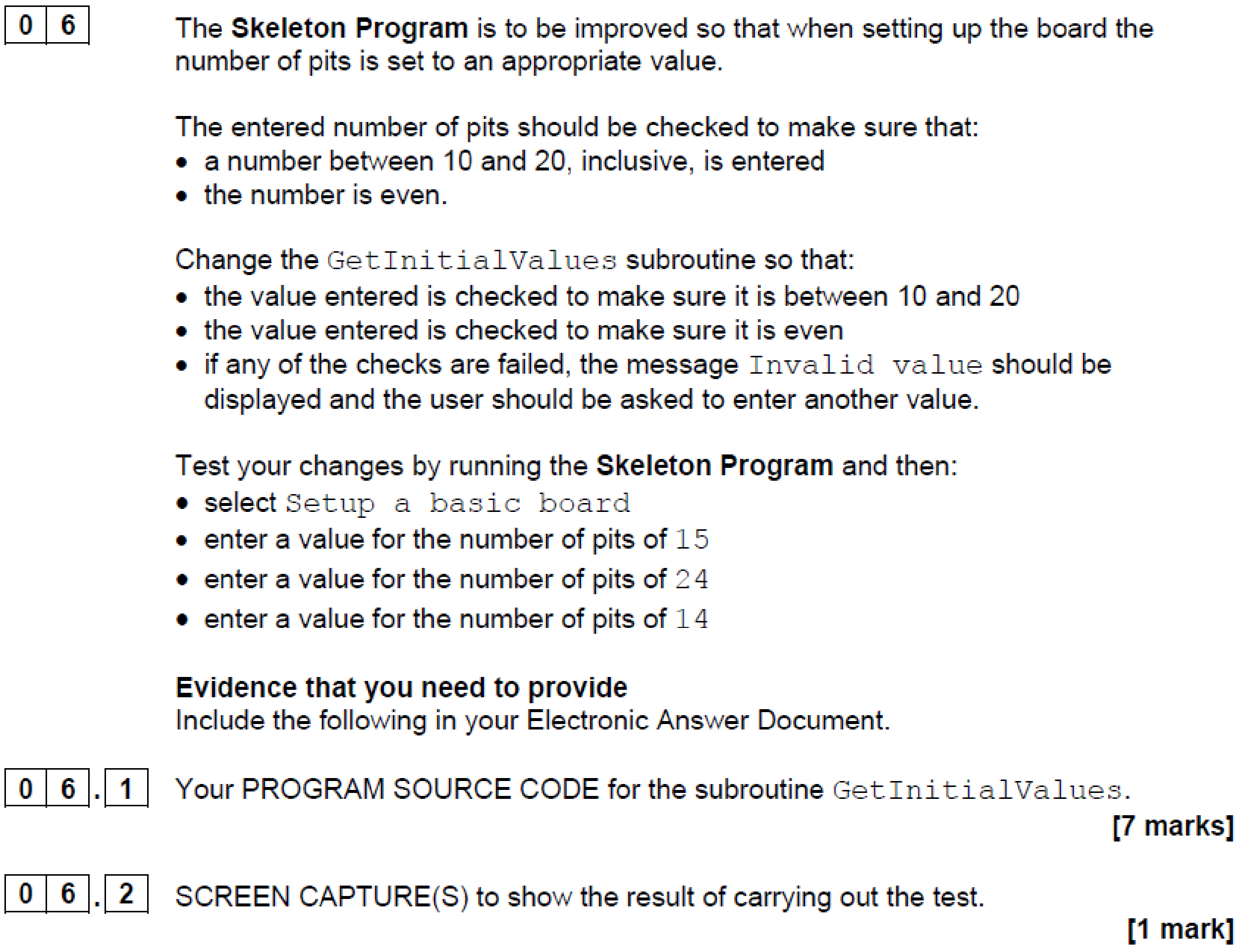
For part 5.1 The student has successfully recognised that a loop is required that repeats until `Board[Seeds]` is greater than zero. They have also displayed the required error message under the correct circumstances.

For part 5.2 it can be seen that the error message is displayed when an invalid value is entered so the mark can be awarded.

Question number 6

Question number 6 required students to add validation to the `GetInitialValues` subroutine to ensure that the entered number of pits is between 10 and 20 and is an even number.

Question number part 6.1 and 6.2



0 6 The **Skeleton Program** is to be improved so that when setting up the board the number of pits is set to an appropriate value.

The entered number of pits should be checked to make sure that:

- a number between 10 and 20, inclusive, is entered
- the number is even.

Change the `GetInitialValues` subroutine so that:

- the value entered is checked to make sure it is between 10 and 20
- the value entered is checked to make sure it is even
- if any of the checks are failed, the message `Invalid value` should be displayed and the user should be asked to enter another value.

Test your changes by running the **Skeleton Program** and then:

- select `Setup a basic board`
- enter a value for the number of pits of `15`
- enter a value for the number of pits of `24`
- enter a value for the number of pits of `14`

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

0 6 . 1 Your PROGRAM SOURCE CODE for the subroutine `GetInitialValues`.

[7 marks]

0 6 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Mark scheme

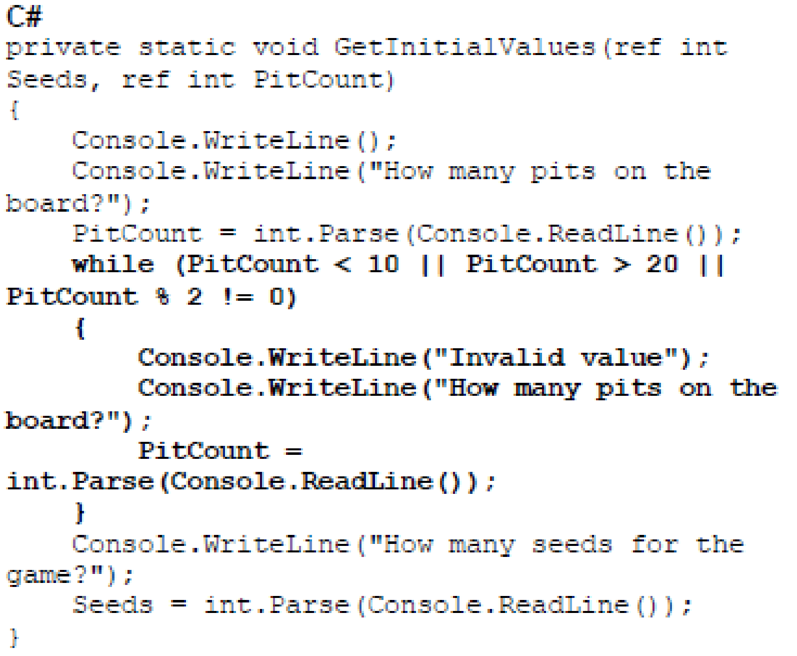
Question	Part	Marking guidance	Total marks
06	1	**1 mark:** Check against lower bound of `PitCount` **1 mark:** Check against upper bound of `PitCount` **1 mark:** Checks for both bounds are correct **1 mark:** Correct test that variable `PitCount` is even **1 mark:** Correctly combine the three checks **1 mark:** Appropriate message and new value for `PitCount` is input and stored correctly **1 mark:** Loop ensures that input is requested again until student's conditions are met **MAX 6 if any errors** **Python** (code below) **VB** (code below)	7 AO3=7

Python

```
def GetInitialValues():
    print()
    PitCount = int(input('How many pits on the board? '))
    while PitCount < 10 or PitCount > 20 or PitCount % 2 != 0:
        print('Invalid value')
        PitCount = int(input('How many pits on the board? '))
    Seeds = int(input('How many seeds for the game? '))
    return PitCount, Seeds
```

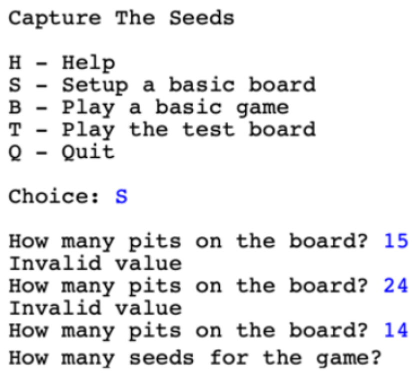
VB

```
Sub GetInitialValues(ByRef Seeds As Integer, ByRef PitCount As Integer)
    Console.WriteLine()
    Console.WriteLine("How many pits on the board?")
    PitCount = Console.ReadLine()
    While PitCount < 10 Or PitCount > 20 Or PitCount Mod 2 <> 0
        Console.WriteLine("Invalid value")
        Console.WriteLine("How many pits on the board?")
        PitCount = Console.ReadLine()
    End While
    Console.WriteLine("How many seeds for the game?")
    Seeds = Console.ReadLine()
End Sub
```



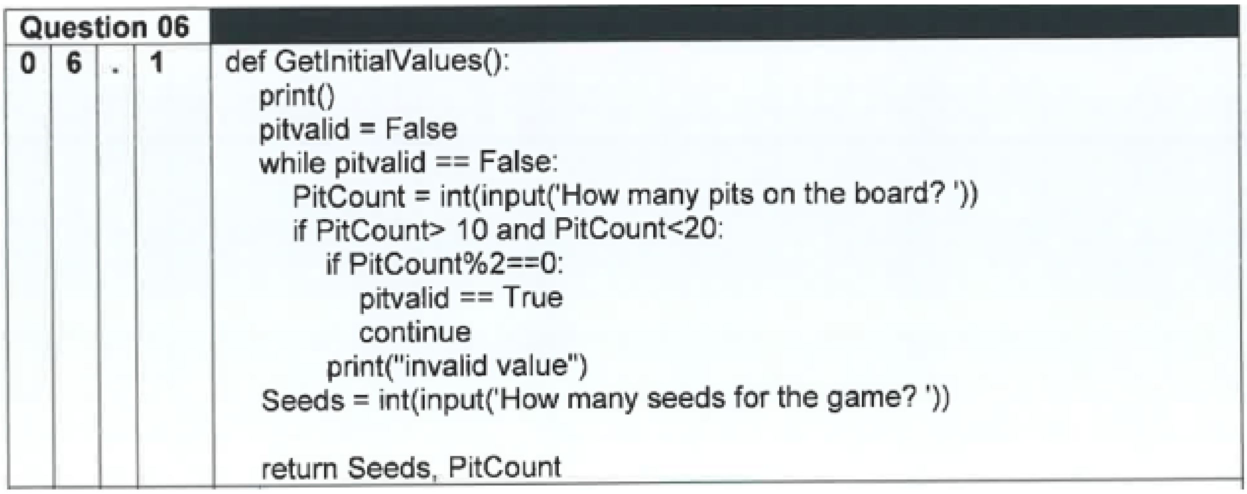
```
C#
private static void GetInitialValues(ref int
Seeds, ref int PitCount)
{
    Console.WriteLine();
    Console.WriteLine("How many pits on the
board?");
    PitCount = int.Parse(Console.ReadLine());
    while (PitCount < 10 || PitCount > 20 ||
PitCount % 2 != 0)
    {
        Console.WriteLine("Invalid value");
        Console.WriteLine("How many pits on the
board?");
        PitCount =
int.Parse(Console.ReadLine());
    }
    Console.WriteLine("How many seeds for the
game?");
    Seeds = int.Parse(Console.ReadLine());
}
```

Question	Part	Marking guidance	Total marks
06	**2**	Screen capture(s) must match code from Q6.1 **1 mark:** Screen capture shows the setup being run with values 15, 24 and then 14 being tested and the error messages.	**1** **AO3=1**



```
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: S

How many pits on the board? 15
Invalid value
How many pits on the board? 24
Invalid value
How many pits on the board? 14
How many seeds for the game?
```

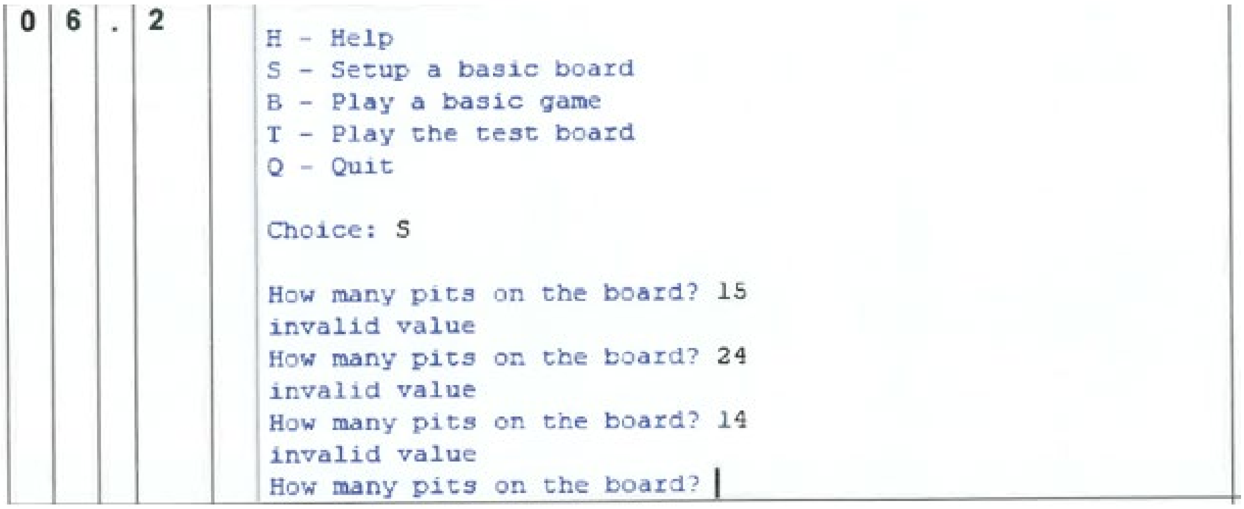
Response A



Question 06

0 6 . 1

```
def GetInitialValues():
    print()
    pitvalid = False
    while pitvalid == False:
        PitCount = int(input('How many pits on the board? '))
        if PitCount> 10 and PitCount<20:
            if PitCount%2==0:
                pitvalid == True
                continue
            print("invalid value")
    Seeds = int(input('How many seeds for the game? '))

    return Seeds, PitCount
```



0 6 . 2

```
H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: S

How many pits on the board? 15
invalid value
How many pits on the board? 24
invalid value
How many pits on the board? 14
invalid value
How many pits on the board? |
```

Commentary

This response received **4 marks** for part **6.1** and **0 marks** for part **6.2**.

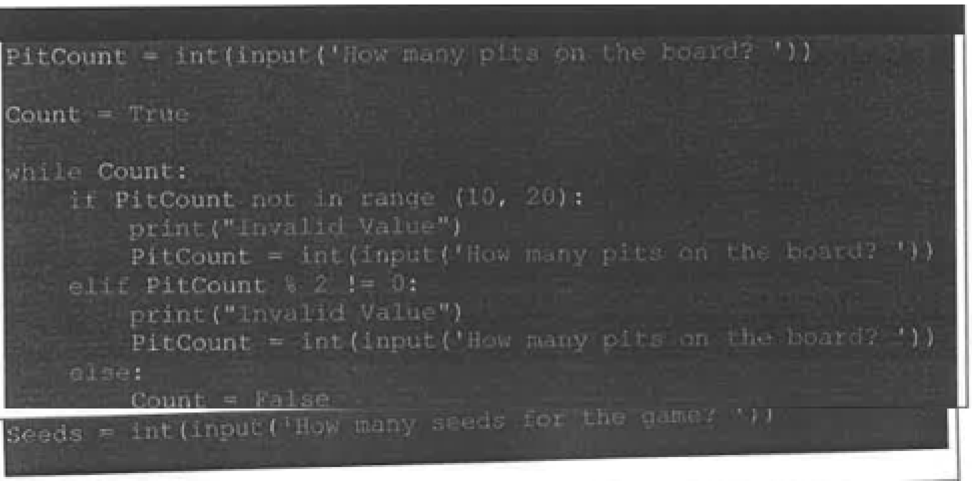
For part 6.1, the student has performed checks against the upper and lower bounds (10 and 20) so the first two marks on the mark scheme were awarded, but the tests are not correct (the conditions should be >= and <=) so the third mark on the mark scheme for the boundary checks being correct cannot be awarded. Students could check their code works for boundary test data to avoid mistakes like this. The check for an even number is correct so the fourth mark point can be awarded. The fifth mark point for combining the checks can also be awarded as the nesting of them combines the three checks correctly, it does not matter that some of the conditions were incorrect for this mark point. The sixth and seventh mark points cannot be awarded. The error message is displayed even when a valid input is made. Additionally, the loop does not work correctly as the student has used == instead of = when assigning `True` to `PitValid`, so the loop will not exit even if the conditions are met.

For part 6.2, no marks are awarded as the tests have been failed by the student's code.

Response B

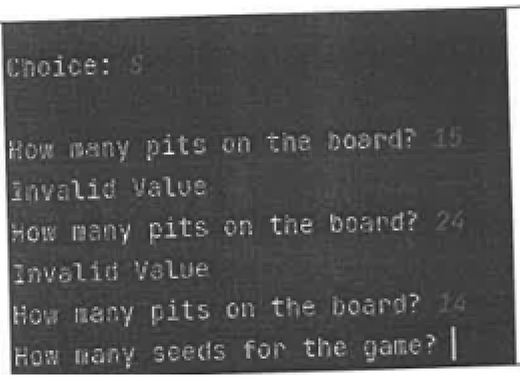
Question 06

0 6 . 1



```
PitCount = int(input('How many pits on the board? '))

Count = True

while Count:
    if PitCount not in range (10, 20):
        print("Invalid Value")
        PitCount = int(input('How many pits on the board? '))
    elif PitCount % 2 != 0:
        print("Invalid Value")
        PitCount = int(input('How many pits on the board? '))
    else:
        Count = False
Seeds = int(input('How many seeds for the game? '))
```

0 6 . 2



```
Choice: 5

How many pits on the board? 15
Invalid Value
How many pits on the board? 24
Invalid Value
How many pits on the board? 14
How many seeds for the game? |
```

Commentary

This response received **6 marks** for part **6.1** and **1 mark** for part **6.2**.

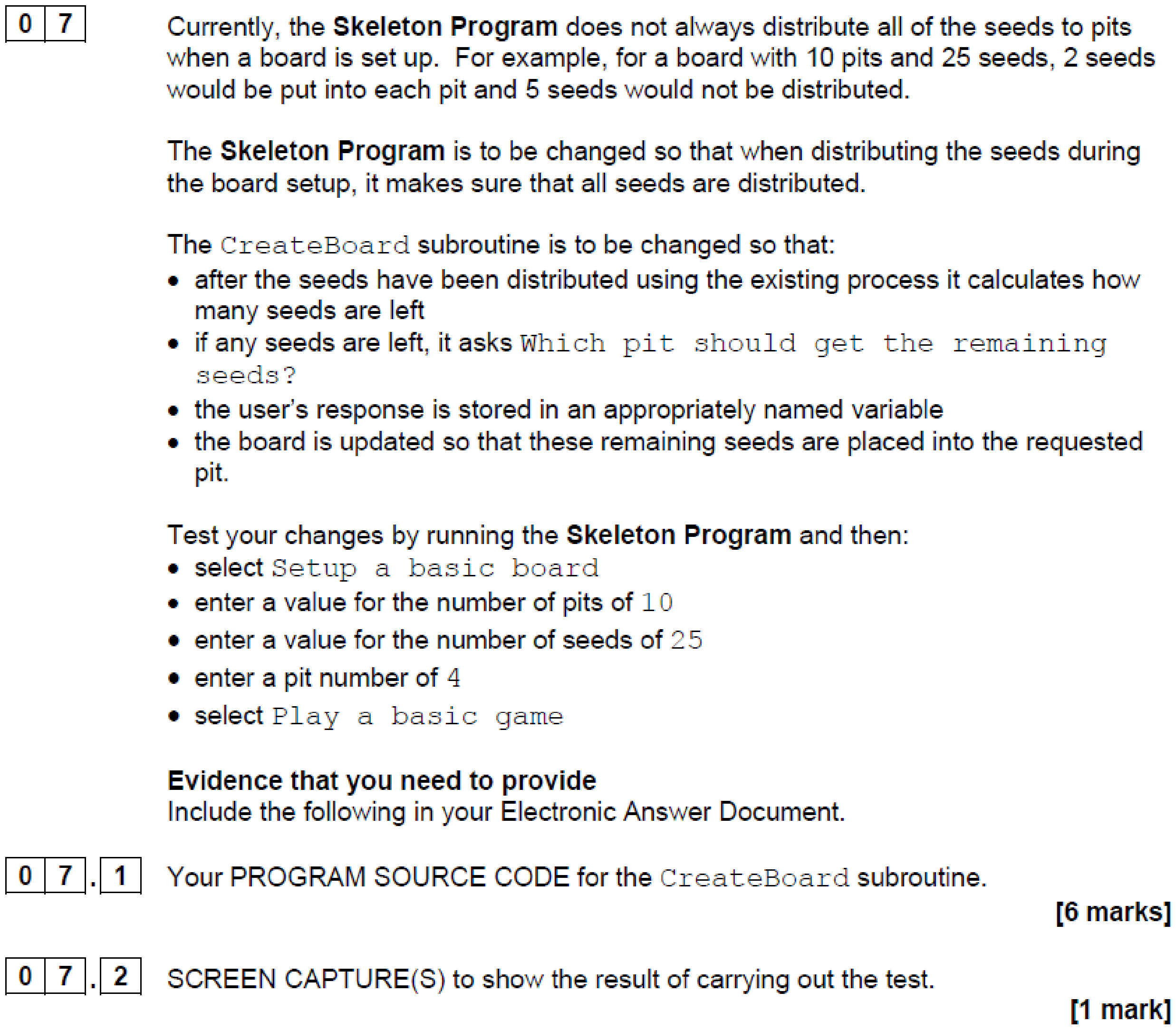
For part 6.1, the solution is almost fully correct. There is a loop that repeats until the conditions are met and the value is re-input when the conditions are not met. The only issue is that the upper boundary of the range is incorrect. Values between 10 and 20 should be accepted. To achieve this, the second value in the `range` command should be 21 rather than 20. Therefore, mark point three from the mark scheme could not be awarded.

For part 6.2, despite the minor error in the code the program worked correctly for the test data so the mark is awarded.

Question number 7

Question number 7 required that students modified the `CreateBoard` subroutine so that all seeds were always distributed into pits at the start of the game.

Question number part 7.1 and 7.2



0 7

Currently, the **Skeleton Program** does not always distribute all of the seeds to pits when a board is set up. For example, for a board with 10 pits and 25 seeds, 2 seeds would be put into each pit and 5 seeds would not be distributed.

The **Skeleton Program** is to be changed so that when distributing the seeds during the board setup, it makes sure that all seeds are distributed.

The `CreateBoard` subroutine is to be changed so that:

- after the seeds have been distributed using the existing process it calculates how many seeds are left
- if any seeds are left, it asks `Which pit should get the remaining seeds?`
- the user's response is stored in an appropriately named variable
- the board is updated so that these remaining seeds are placed into the requested pit.

Test your changes by running the **Skeleton Program** and then:

- select `Setup a basic board`
- enter a value for the number of pits of `10`
- enter a value for the number of seeds of `25`
- enter a pit number of `4`
- select `Play a basic game`

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

0 7 . 1 Your PROGRAM SOURCE CODE for the `CreateBoard` subroutine.

[6 marks]

0 7 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks
07	1	**1 mark:** Use of remainder operator or equivalent **1 mark:** Correctly calculate number of remaining seeds **1 mark:** Selection statement which compares calculated integer value with 0. **A.** compare with 1 if greater than or equal to relational operator has been used **1 mark:** Appropriate prompt for remaining seeds **1 mark:** Response stored in an appropriate variable **1 mark:** Board updated appropriately **MAX 5 if any errors**	6 AO3=6

Python

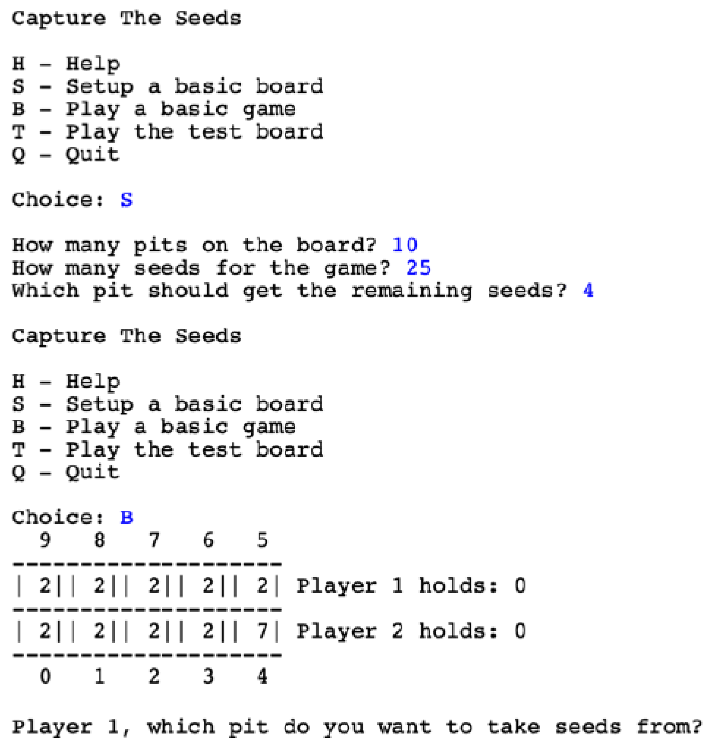
```
def CreateBoard(Seeds, PitCount):
    global Board
    for Count in range(PitCount):
        Board.append(Seeds // PitCount)
    Remainder = Seeds % PitCount
    if Remainder > 0:
        Pit = int(input('Which pit should get the remaining seeds? '))
        Board[Pit] = Board[Pit] + Remainder
```

VB

```
Sub CreateBoard(Seeds As Integer, PitCount As Integer)
    ReDim Board(PitCount - 1)
    For Count = 0 To PitCount - 1
        Board(Count) = Seeds \ PitCount
    Next
    Dim Remainder As Integer = Seeds Mod PitCount
    If Remainder > 0 Then
        Console.WriteLine("Which pit should get the remaining seed? ")
        Dim Pit As Integer = Console.ReadLine()
        Board(Pit) = Board(Pit) + Remainder
    End If
End Sub
```

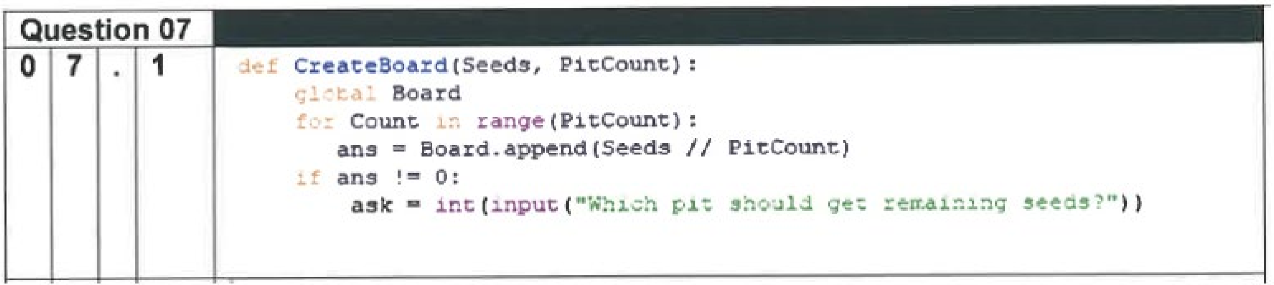
C#

```
private static void CreateBoard(int Seeds, int
PitCount)
{
    Board = new int[PitCount];
    for (int Count = 0; Count < PitCount;
Count++)
    {
        Board[Count] = Seeds / PitCount;
    }
    int Remainder = Seeds % PitCount;
    if (Remainder > 0)
    {
        Console.WriteLine("Which pit should get
the remaining seeds? ");
        int Pit =
int.Parse(Console.ReadLine());
        Board[Pit] = Board[Pit] + Remainder;
    }
}
```

Question	Part	Marking guidance	Total marks
07	2	Screen capture(s) must match code from Q7.1 **1 mark:** Screen capture shows the setup being run with values 10, 25 and then 4 being entered. A basic game should be started showing the board with extra seeds in pit 4. 	1 **AO3=1**

```
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: S

How many pits on the board? 10
How many seeds for the game? 25
Which pit should get the remaining seeds? 4

Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: B
   9    8    7    6    5
---------------------
|  2||  2||  2||  2||  2| Player 1 holds: 0
---------------------
|  2||  2||  2||  2||  7| Player 2 holds: 0
---------------------
   0    1    2    3    4

Player 1, which pit do you want to take seeds from?
```

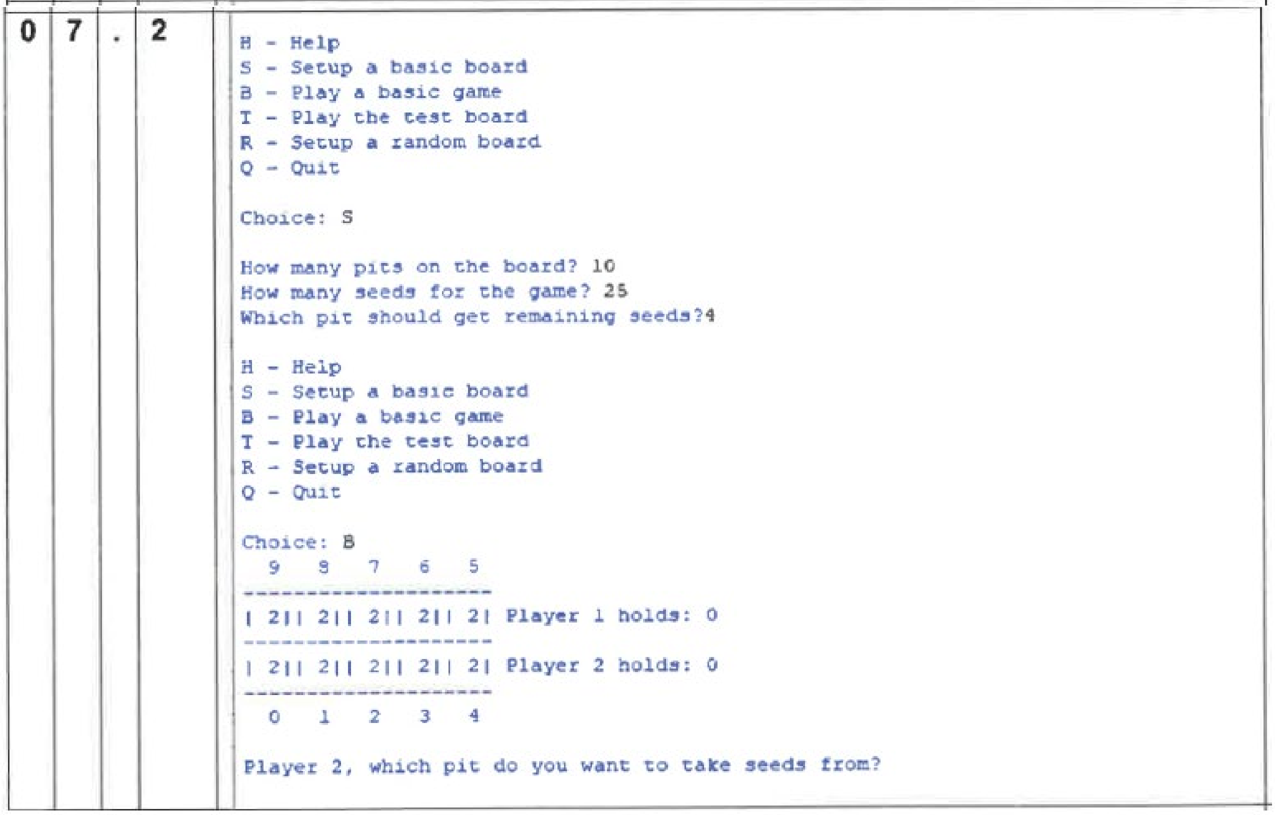
Response A



Question 07

0 7 . 1

```
def CreateBoard(Seeds, PitCount):
    global Board
    for Count in range(PitCount):
       ans = Board.append(Seeds // PitCount)
    if ans != 0:
        ask = int(input("Which pit should get remaining seeds?"))
```



0 7 . 2

```
H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
R - Setup a random board
Q - Quit

Choice: S

How many pits on the board? 10
How many seeds for the game? 25
Which pit should get remaining seeds?4

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
R - Setup a random board
Q - Quit

Choice: B
  9    8    7    6    5
--------------------
| 2|| 2|| 2|| 2|| 2| Player 1 holds: 0
--------------------
| 2|| 2|| 2|| 2|| 2| Player 2 holds: 0
--------------------
  0    1    2    3    4

Player 2, which pit do you want to take seeds from?
```

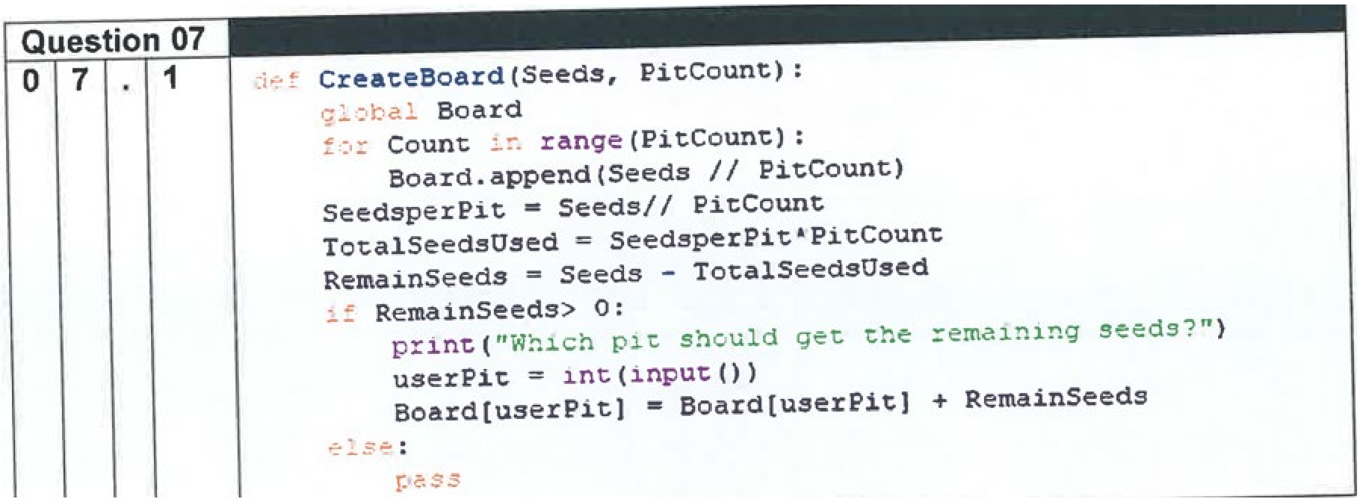
Commentary

This response received **2 marks** for part **7.1** and **0 marks** for part **7.2**.

For part 7.1, the student has not correctly worked out the number of remaining seeds or updated the board, so mark points one to three and six cannot be awarded. Mark points four and five can be awarded as the student has prompted the user to input where to put the remaining seeds and has correctly stored this value in a variable. This student has demonstrated good exam technique as despite the fact that they could not do the earlier part of the question correctly they have still attempted the later part and so have achieved some marks.

The code did not work out the number of remaining seeds so no marks could be awarded for the test result in part 7.2.

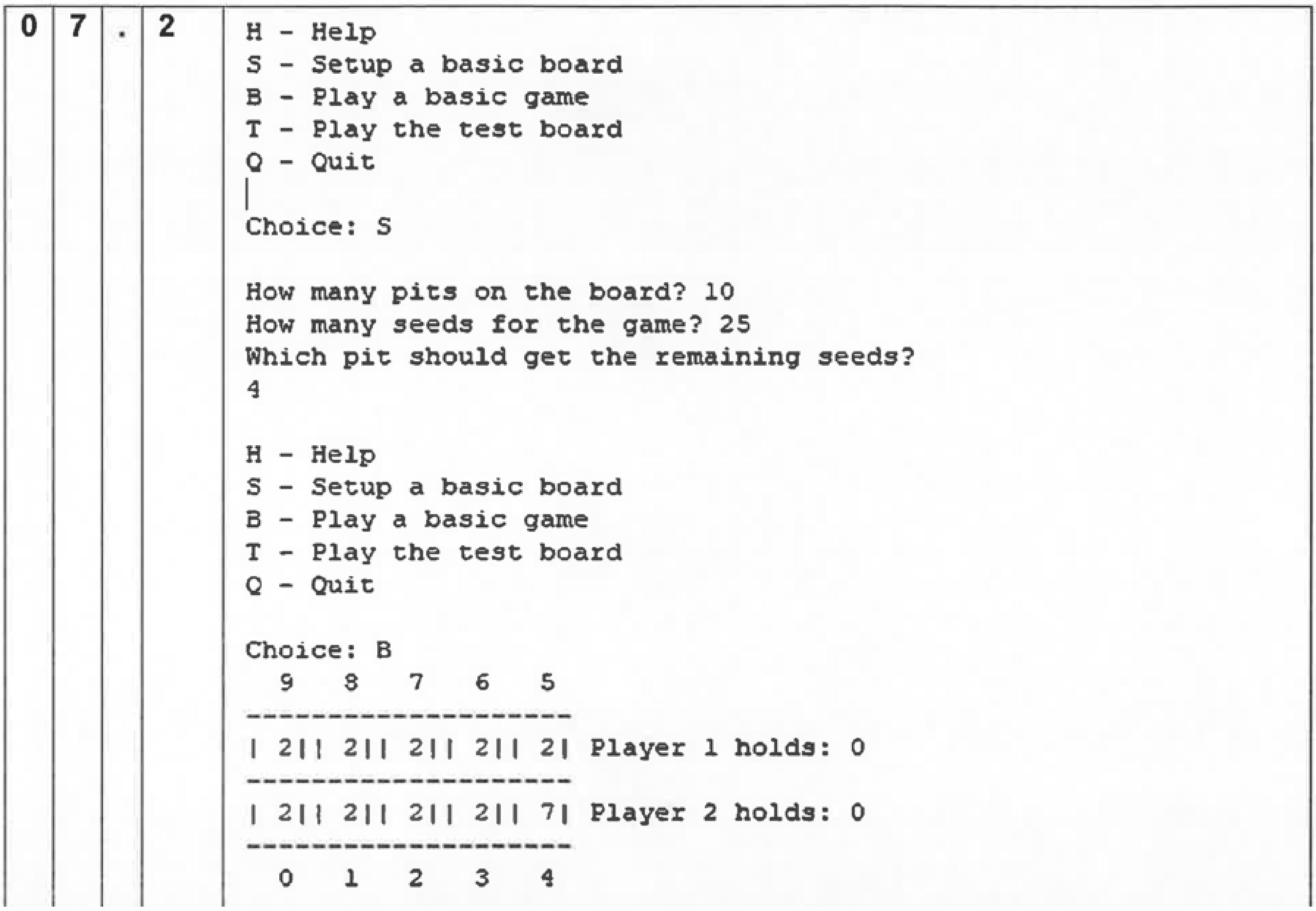
Response B



Question 07

0 7 . 1

```
def CreateBoard(Seeds, PitCount):
    global Board
    for Count in range(PitCount):
        Board.append(Seeds // PitCount)
    SeedsperPit = Seeds// PitCount
    TotalSeedsUsed = SeedsperPit*PitCount
    RemainSeeds = Seeds - TotalSeedsUsed
    if RemainSeeds> 0:
        print("Which pit should get the remaining seeds?")
        userPit = int(input())
        Board[userPit] = Board[userPit] + RemainSeeds
    else:
        pass
```



0 7 . 2

```
H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit
|
Choice: S

How many pits on the board? 10
How many seeds for the game? 25
Which pit should get the remaining seeds?
4

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: B
  9   8   7   6   5
--------------------
| 2|| 2|| 2|| 2|| 2| Player 1 holds: 0
--------------------
| 2|| 2|| 2|| 2|| 7| Player 2 holds: 0
--------------------
  0   1   2   3   4
```

Commentary

This was a very good response that received **6 marks** for part **7.1** and **1 mark** for part **7.2**.

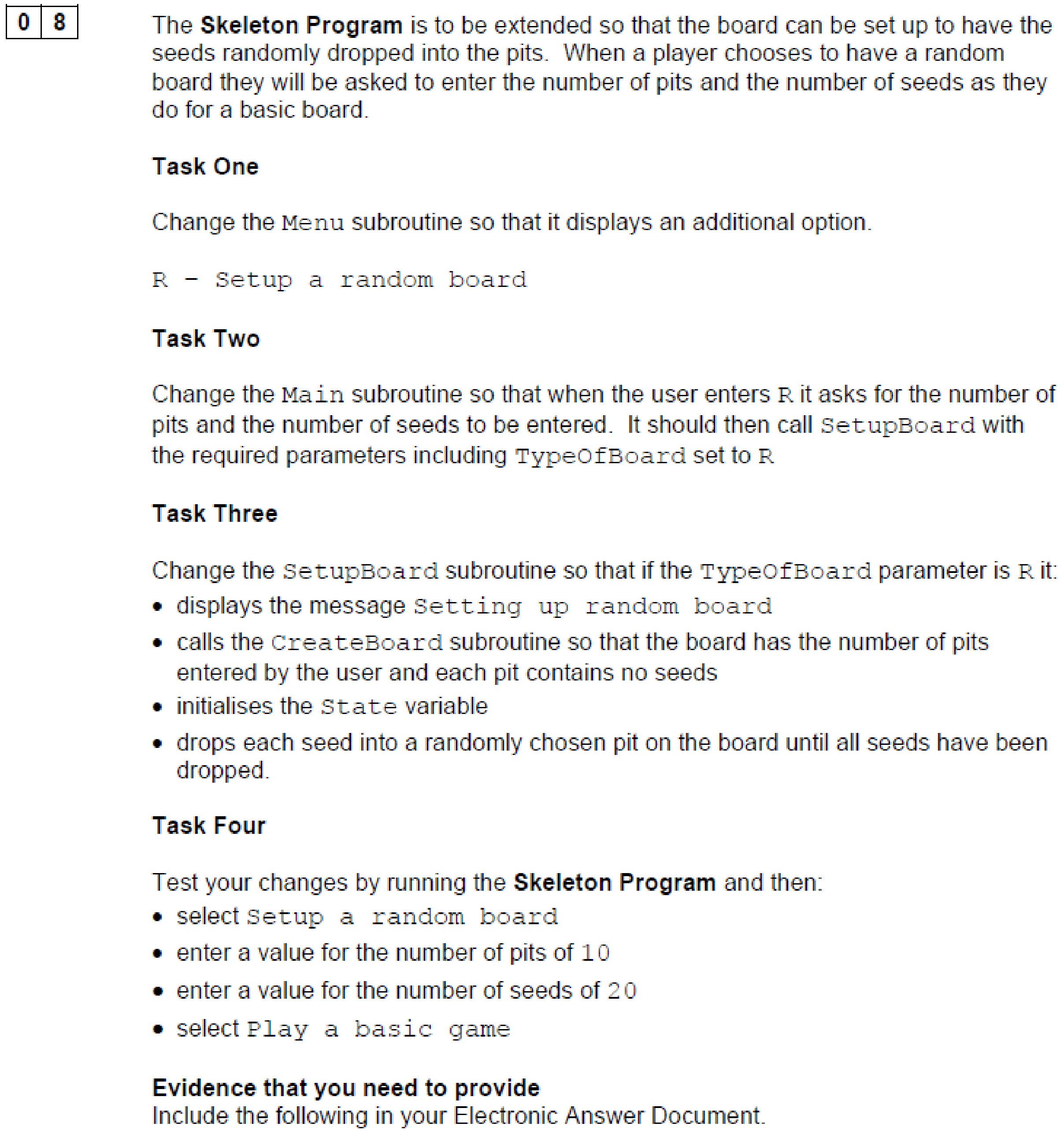
For part 7.1, the student has not used the approach shown in the mark scheme to calculate the number of remaining seeds. Instead of using the MOD operator, they have calculated how many seeds were put into pits using the original method and subtracted this from the number of seeds in the game to work out how many seeds are left. Alternative valid responses will always be credited, so the student received full marks for this question part.

For part 7.2 the program has performed correctly when the test data was input so the mark was awarded.

Question number 8

Question number 8 was a more complex programming question which required that students added a new option to the Skeleton Program to setup a random board.

Question number part 8.1, 8.2, 8.3 and 8.4



0 8

The **Skeleton Program** is to be extended so that the board can be set up to have the seeds randomly dropped into the pits. When a player chooses to have a random board they will be asked to enter the number of pits and the number of seeds as they do for a basic board.

Task One

Change the `Menu` subroutine so that it displays an additional option.

```
R – Setup a random board
```

Task Two

Change the `Main` subroutine so that when the user enters `R` it asks for the number of pits and the number of seeds to be entered. It should then call `SetupBoard` with the required parameters including `TypeOfBoard` set to `R`

Task Three

Change the `SetupBoard` subroutine so that if the `TypeOfBoard` parameter is `R` it:

- displays the message `Setting up random board`
- calls the `CreateBoard` subroutine so that the board has the number of pits entered by the user and each pit contains no seeds
- initialises the `State` variable
- drops each seed into a randomly chosen pit on the board until all seeds have been dropped.

Task Four

Test your changes by running the **Skeleton Program** and then:

- select `Setup a random board`
- enter a value for the number of pits of `10`
- enter a value for the number of seeds of `20`
- select `Play a basic game`

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

0 8 . 1 Your PROGRAM SOURCE CODE for the `Menu` subroutine.

[1 mark]

0 8 . 2 Your PROGRAM SOURCE CODE for the `Main` subroutine.

[4 marks]

0 8 . 3 Your PROGRAM SOURCE CODE for the `SetupBoard` subroutine.

[10 marks]

0 8 . 4 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks
08	1	**1 mark:** Option for setup random board added into menu options **I.** Ordering of options **A.** Minor typos in output message	1 AO3=1

Python

```
def Menu():
    print()
    print('Capture The Seeds')
    print()
    print('H - Help')
    print('S - Setup a basic board')
    print('R - Setup a random board')
    print('B - Play a basic game')
    print('T - Play the test board')
    print('Q - Quit')
    print()
```

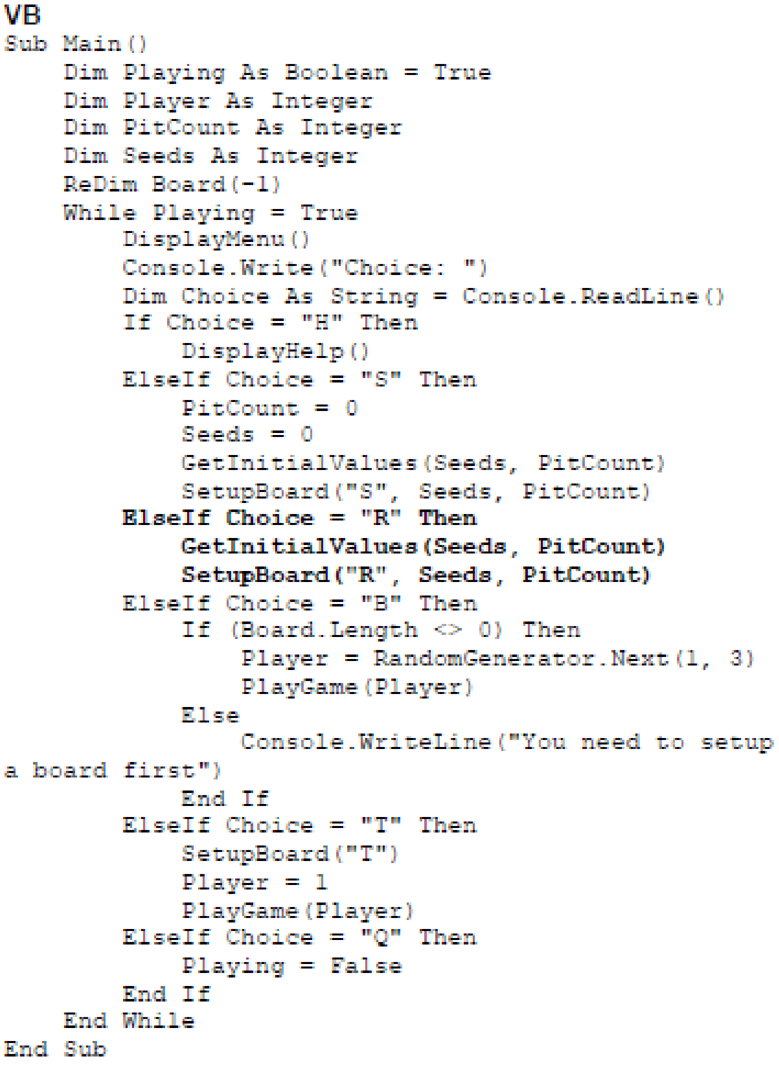
VB

```
Sub DisplayMenu()
    Console.WriteLine()
    Console.WriteLine("Capture The Seeds")
    Console.WriteLine()
    Console.WriteLine("H - Help")
    Console.WriteLine("S - Setup a basic board")
    Console.WriteLine("R - Setup a random board")
    Console.WriteLine("B - Play a basic game")
    Console.WriteLine("T - Play the test board")
    Console.WriteLine("Q - Quit")
    Console.WriteLine()
End Sub
```

		C#
		private static void DisplayMenu()
		{
		    Console.WriteLine();
		    Console.WriteLine("Capture The Seeds");
		    Console.WriteLine();
		    Console.WriteLine("H – Help");
		    Console.WriteLine("S – Setup a basic board");
		    Console.WriteLine("R – Setup a random board");
		    Console.WriteLine("B – Play a basic game");
		    Console.WriteLine("T – Play the test board");
		    Console.WriteLine("Q – Quit");
		    Console.WriteLine();
		}	

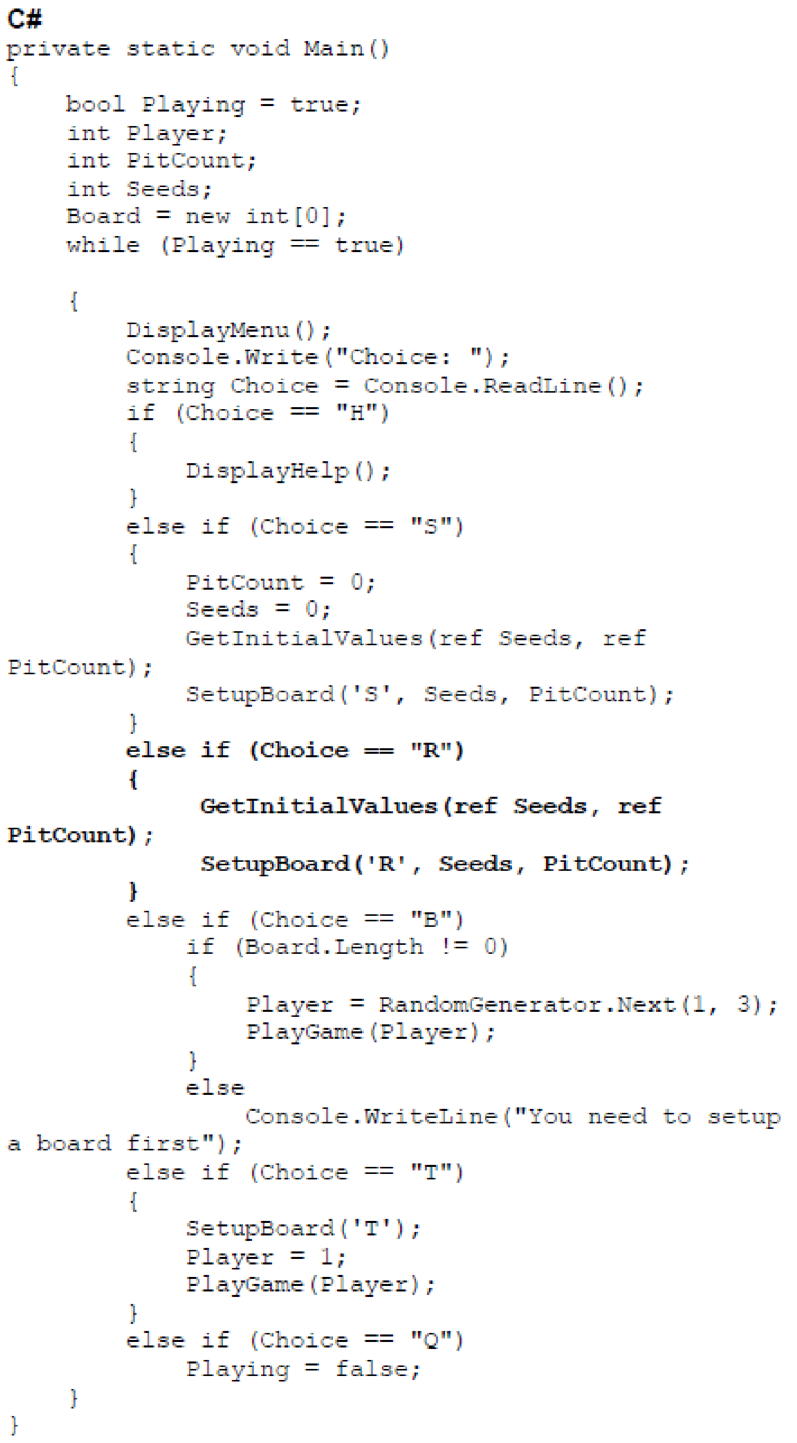
Question	Part	Marking guidance	Total marks
08	2	**1 mark:** Extra case added for selection statement **1 mark:** Checks `Choice` = `'R'` **1 mark:** Makes appropriate call to `GetInitialValues` or equivalent **1 mark:** makes appropriate call to `SetupBoard` **MAX 3 if any errors** **Python**	4 AO3=4

```
def Main():
    Playing = True
    while Playing:
        DisplayMenu()
        Choice = input('Choice: ')
        if Choice == 'H':
            DisplayHelp()
        elif Choice == 'S':
            PitCount, Seeds = GetInitialValues ()
            SetupBoard('S', PitCount, Seeds)
        elif Choice == 'R':
            Seeds, PitCount = GetInitialValues ()
            SetupBoard('R', Seeds, PitCount)
        elif Choice == 'B':
            if len(Board) != 0:
                Player = random.randint(1,2)
                PlayGame(Player)
            else:
                print('You need to setup a board
first')
        elif Choice == 'T':
            SetupBoard('T')
            Player = 1
            PlayGame(Player)
        elif Choice == 'Q':
            Playing = False
```



VB

```
Sub Main()
    Dim Playing As Boolean = True
    Dim Player As Integer
    Dim PitCount As Integer
    Dim Seeds As Integer
    ReDim Board(-1)
    While Playing = True
        DisplayMenu()
        Console.Write("Choice: ")
        Dim Choice As String = Console.ReadLine()
        If Choice = "H" Then
            DisplayHelp()
        ElseIf Choice = "S" Then
            PitCount = 0
            Seeds = 0
            GetInitialValues(Seeds, PitCount)
            SetupBoard("S", Seeds, PitCount)
        ElseIf Choice = "R" Then
            GetInitialValues(Seeds, PitCount)
            SetupBoard("R", Seeds, PitCount)
        ElseIf Choice = "B" Then
            If (Board.Length <> 0) Then
                Player = RandomGenerator.Next(1, 3)
                PlayGame(Player)
            Else
                Console.WriteLine("You need to setup
a board first")
            End If
        ElseIf Choice = "T" Then
            SetupBoard("T")
            Player = 1
            PlayGame(Player)
        ElseIf Choice = "Q" Then
            Playing = False
        End If
    End While
End Sub
```



C#

```
private static void Main()
{
    bool Playing = true;
    int Player;
    int PitCount;
    int Seeds;
    Board = new int[0];
    while (Playing == true)

    {
        DisplayMenu();
        Console.Write("Choice: ");
        string Choice = Console.ReadLine();
        if (Choice == "H")
        {
            DisplayHelp();
        }
        else if (Choice == "S")
        {
            PitCount = 0;
            Seeds = 0;
            GetInitialValues(ref Seeds, ref
PitCount);
            SetupBoard('S', Seeds, PitCount);
        }
        else if (Choice == "R")
        {
             GetInitialValues(ref Seeds, ref
PitCount);
             SetupBoard('R', Seeds, PitCount);
        }
        else if (Choice == "B")
            if (Board.Length != 0)
            {
                Player = RandomGenerator.Next(1, 3);
                PlayGame(Player);
            }
            else
                Console.WriteLine("You need to setup
a board first");
        else if (Choice == "T")
        {
            SetupBoard('T');
            Player = 1;
            PlayGame(Player);
        }
        else if (Choice == "Q")
            Playing = false;
    }
}
```

Question	Part	Marking guidance	Total marks
08	3	**1 mark:** extension to selection statement for `TypeOfBoard` equal to `R` **1 mark:** Displays message `Setting up random board` **1 mark:** Call to `CreateBoard` **1 mark:** Correct parameters used for call to `CreateBoard` **1 mark:** Correctly setting up `State` variable **1 mark:** Generating a random number **1 mark:** Random number within the correct range **1 mark:** Correctly indexing `Board` with the random number **1 mark:** Adding 1 to current value of indexed board position **1 mark:** Iteration controlled correctly by number of seeds remaining MAX 9 if any errors Python	10 AO3=10

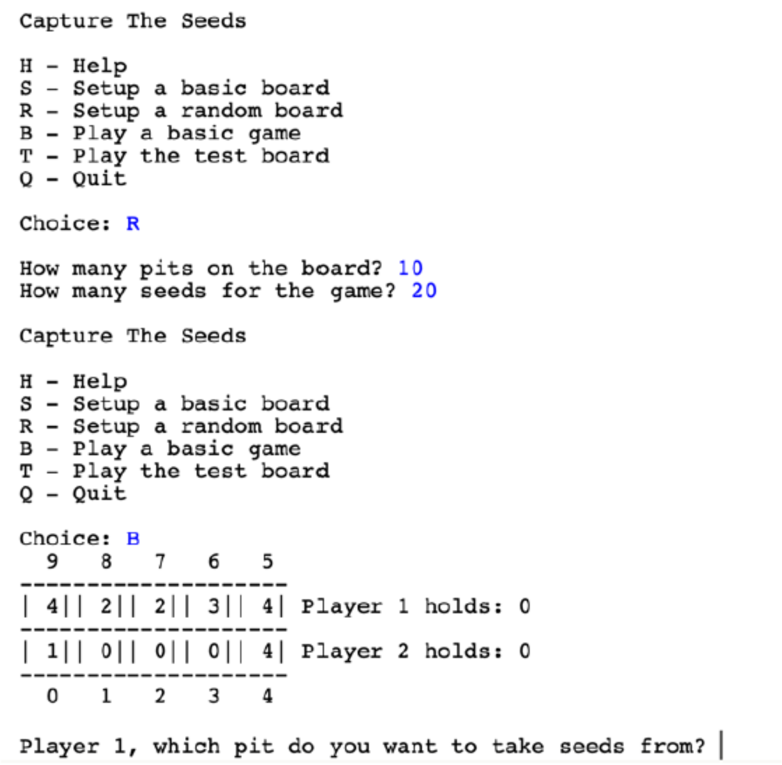
```
    elif TypeOfBoard == 'S':
        CreateBoard(Seeds, PitCount)
        State=[Seeds,0,0]
    elif TypeOfBoard == 'R':
        print('Setting up random board')
        CreateBoard(0, PitCount)
        State = [Seeds, 0, 0]
        while Seeds > 0:
            RandomPit = random.randint(0, PitCount - 1)
            Board[RandomPit] = Board[RandomPit] + 1
            Seeds = Seeds - 1
```

VB

```
    If TypeOfBoard = "T" Then
        Board = {1, 0, 3, 1, 2, 2, 0, 0, 0,
0, 2, 1}
        State = {48, 16, 20}
    ElseIf TypeOfBoard = "R" Then
        Console.WriteLine("Setting up
random board")
        CreateBoard(0, PitCount)
        State = {Seeds, 0 ,0}
        While Seeds > 0
            Dim RandomPit As Integer =
RandomGenerator.Next(0, PitCount)
            Board(RandomPit) =
Board(RandomPit) + 1
            Seeds = Seeds - 1
        End While
    ElseIf TypeOfBoard = "S" Then
        CreateBoard(Seeds, PitCount)
        State = {Seeds, 0, 0}
    End If
```

C#

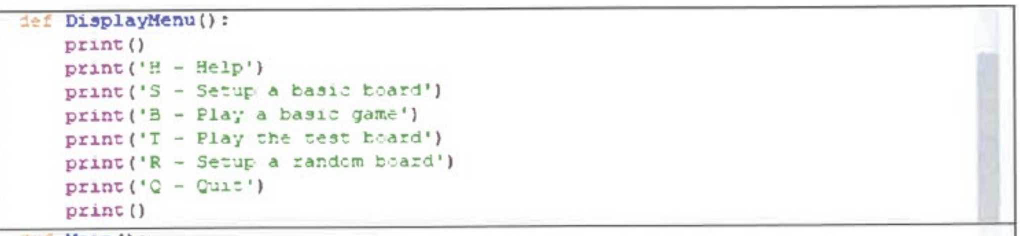
```
    if (TypeOfBoard == 'T')
    {
        Board = new int[12] { 1, 0, 3, 1,
2, 2, 0, 0, 0, 0, 2, 1 };
        State = new int[3] { 48, 16, 20 };
    }
    else if (TypeOfBoard == 'R')
    {
        Console.WriteLine("Setting up
random board");
        CreateBoard(0, PitCount);
        State = new int[3] { Seeds, 0, 0};
        while (Seeds > 0)
        {
            int RandomPit =
RandomGenerator.Next(0, PitCount);
            Board[RandomPit] =
Board[RandomPit] + 1
            Seeds = Seeds - 1
        }
    }
    else if (TypeOfBoard == 'S')
    {
        CreateBoard(Seeds, PitCount);
        State = new int[3] { Seeds, 0, 0 };
    }
```

Question	Part	Marking guidance	Total marks
08	**4**	Screen capture(s) must match code from Q8.1, Q8.2 and Q8.3. **1 mark:** Screen capture shows the random board setup being called with the values 10 and 20 being entered. A board should then be displayed with random distribution of the seeds. Total of the seeds should be 20. 	**1** **AO3=1**

```
Capture The Seeds

H - Help
S - Setup a basic board
R - Setup a random board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: R

How many pits on the board? 10
How many seeds for the game? 20

Capture The Seeds

H - Help
S - Setup a basic board
R - Setup a random board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: B
  9   8   7   6   5
--------------------
| 4|| 2|| 2|| 3|| 4| Player 1 holds: 0
--------------------
| 1|| 0|| 0|| 0|| 4| Player 2 holds: 0
--------------------
  0   1   2   3   4

Player 1, which pit do you want to take seeds from?
```

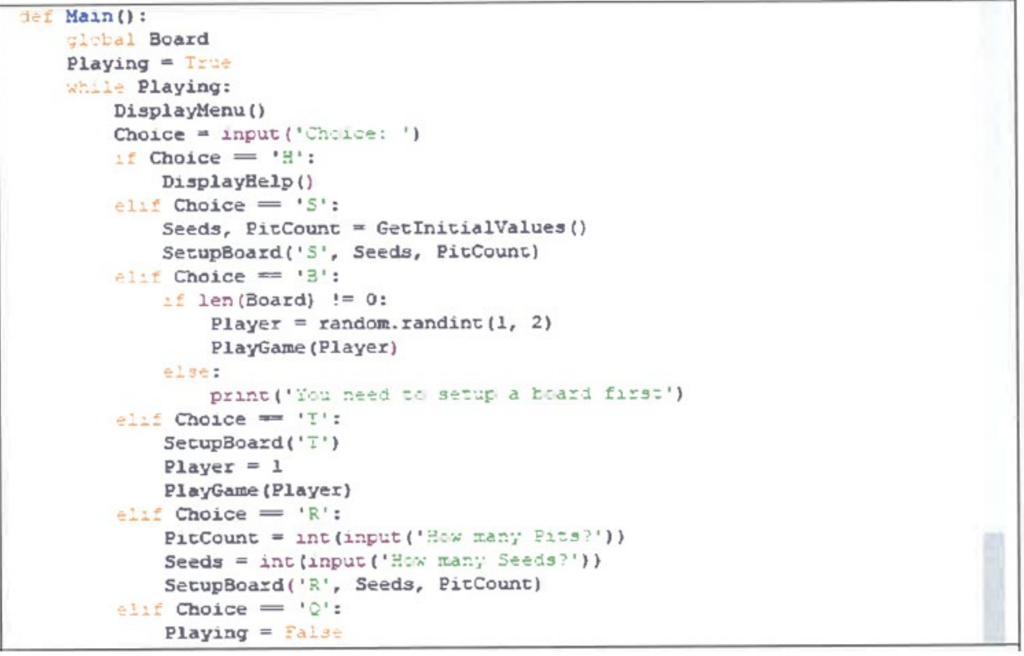
Response A

0 8 . 1



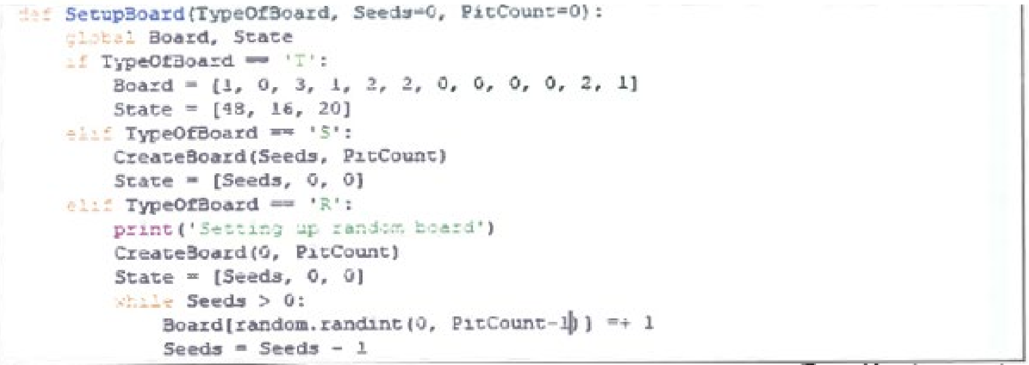
```
def DisplayMenu():
    print()
    print('H - Help')
    print('S - Setup a basic board')
    print('B - Play a basic game')
    print('T - Play the test board')
    print('R - Setup a random board')
    print('Q - Quit')
    print()
```

0 8 . 2



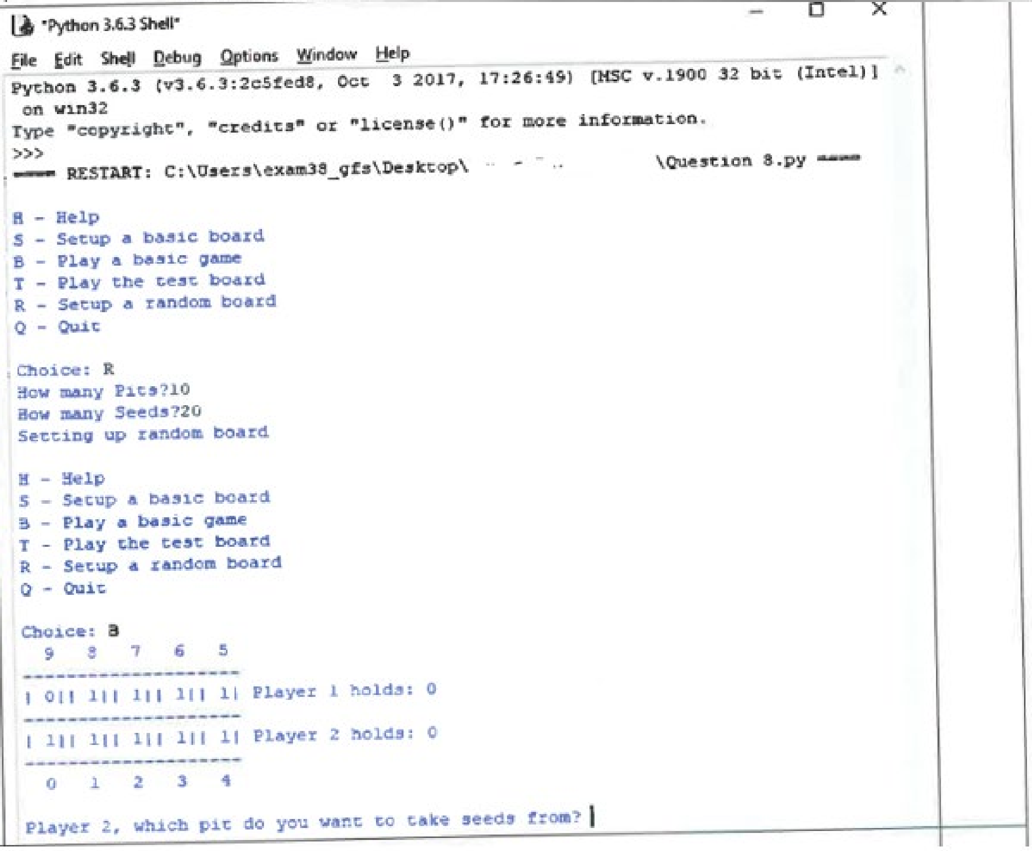
```
def Main():
    global Board
    Playing = True
    while Playing:
        DisplayMenu()
        Choice = input('Choice: ')
        if Choice == 'H':
            DisplayHelp()
        elif Choice == 'S':
            Seeds, PitCount = GetInitialValues()
            SetupBoard('S', Seeds, PitCount)
        elif Choice == 'B':
            if len(Board) != 0:
                Player = random.randint(1, 2)
                PlayGame(Player)
            else:
                print('You need to setup a board first')
        elif Choice == 'T':
            SetupBoard('T')
            Player = 1
            PlayGame(Player)
        elif Choice == 'R':
            PitCount = int(input('How many Pits?'))
            Seeds = int(input('How many Seeds?'))
            SetupBoard('R', Seeds, PitCount)
        elif Choice == 'Q':
            Playing = False
```

0 8 . 3



```
def SetupBoard(TypeOfBoard, Seeds=0, PitCount=0):
    global Board, State
    if TypeOfBoard == 'T':
        Board = [1, 0, 3, 1, 2, 2, 0, 0, 0, 0, 2, 1]
        State = [48, 16, 20]
    elif TypeOfBoard == 'S':
        CreateBoard(Seeds, PitCount)
        State = [Seeds, 0, 0]
    elif TypeOfBoard == 'R':
        print('Setting up random board')
        CreateBoard(0, PitCount)
        State = [Seeds, 0, 0]
        while Seeds > 0:
            Board[random.randint(0, PitCount-1)] =+ 1
            Seeds = Seeds - 1
```

0 8 . 4



```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\exam38_gfs\Desktop\ ..  - ‾ ..        \Question 8.py ====

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
R - Setup a random board
Q - Quit

Choice: R
How many Pits?10
How many Seeds?20
Setting up random board

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
R - Setup a random board
Q - Quit

Choice: B
  9   8   7   6   5
---------------------
| 0|| 1|| 1|| 1|| 1| Player 1 holds: 0
---------------------
| 1|| 1|| 1|| 1|| 1| Player 2 holds: 0
---------------------
  0   1   2   3   4

Player 2, which pit do you want to take seeds from?
```

Commentary

This was a very good response that received **1 mark** for part **8.1**, **4 marks** for part **8.2**, **9 marks** for part **8.3** and **0 marks** for part **8.4**.

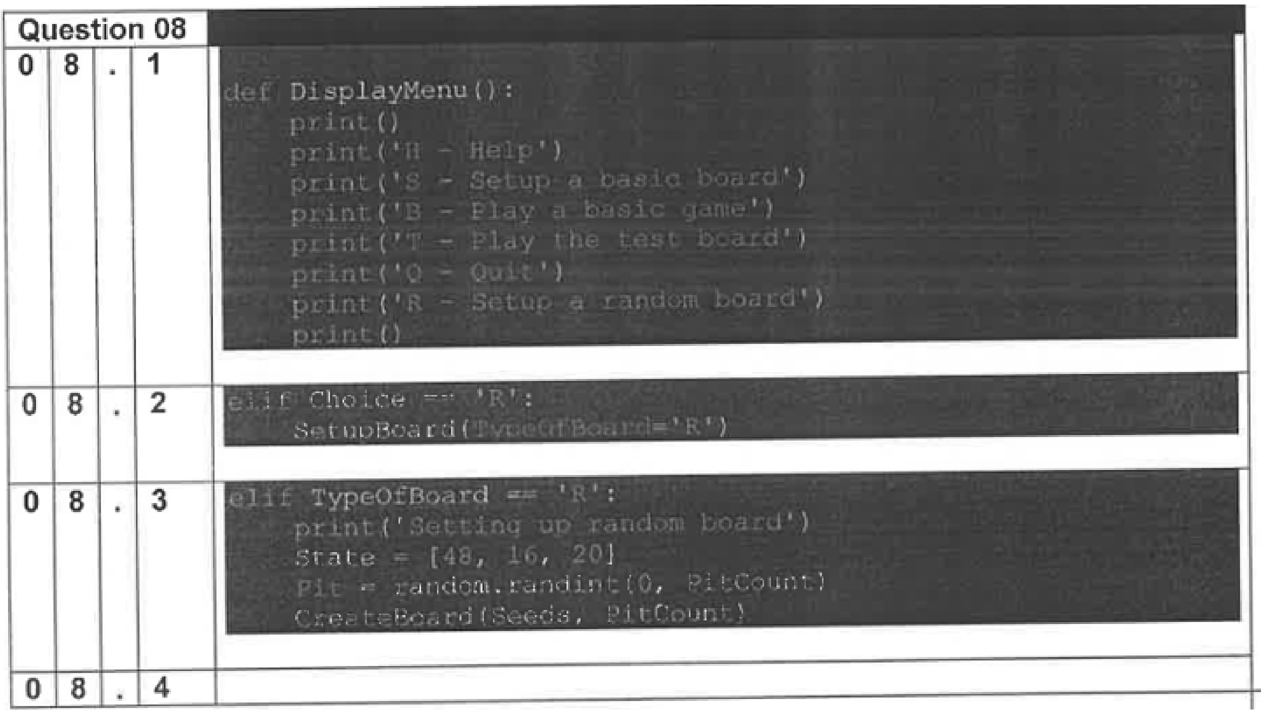
For part 8.1, the student successfully added the new option to the `Menu` subroutine so was awarded the mark.

For part 8.2, the student received all four marks. They have not called the `GetInitialValues` subroutine to input the number of pits and seeds, but mark point three on the mark scheme was still awarded as the student had written their own input commands which achieved the same purpose as the call would have done.

For part 8.3, the student produced a nearly fully working solution, achieving nine of the ten available marks. The only mistake that the student has made is in the method by which they attempt to increment the number of seeds in the pit. The student has written =+1 which will set the number of seeds to one instead of increasing it by one.

For part 8.4, no marks were awarded as the test did not work correctly as a result of the coding error made in part 8.3.

Response B



Question 08

0 8 . 1

```
def DisplayMenu():
    print()
    print('H - Help')
    print('S - Setup a basic board')
    print('B - Play a basic game')
    print('T - Play the test board')
    print('Q - Quit')
    print('R - Setup a random board')
    print()
```

0 8 . 2

```
elif Choice == 'R':
    SetupBoard(TypeOfBoard='R')
```

0 8 . 3

```
elif TypeOfBoard == 'R':
    print('Setting up random board')
    State = [48, 16, 20]
    Pit = random.randint(0, PitCount)
    CreateBoard(Seeds, PitCount)
```

0 8 . 4

Commentary

This response received **1 mark** for part **8.1**, **1 mark** for part **8.2**, **3 marks** for part **8.3** and **0 marks** for part **8.4**.

For part 8.1, the student successfully added the new option to the Menu subroutine so was awarded the mark.

For part 8.2, the student appears to have added an additional alternative to the if command, to check if Choice is equal to R. Inside this alternative, the subroutine SetupBoard is called. The only mark awarded for this response was mark point two from the mark scheme, for comparing Choice to R. Mark point one was not awarded as the student did not show enough of their code for it to be clear that the elif was added to the correct if statement. It is important that students always include the code for the entire subroutine, so that an examiner can see where new code was added. Mark point four was also not awarded because whilst SetupBoard was called, the call was not appropriate because the parameters are incorrect.

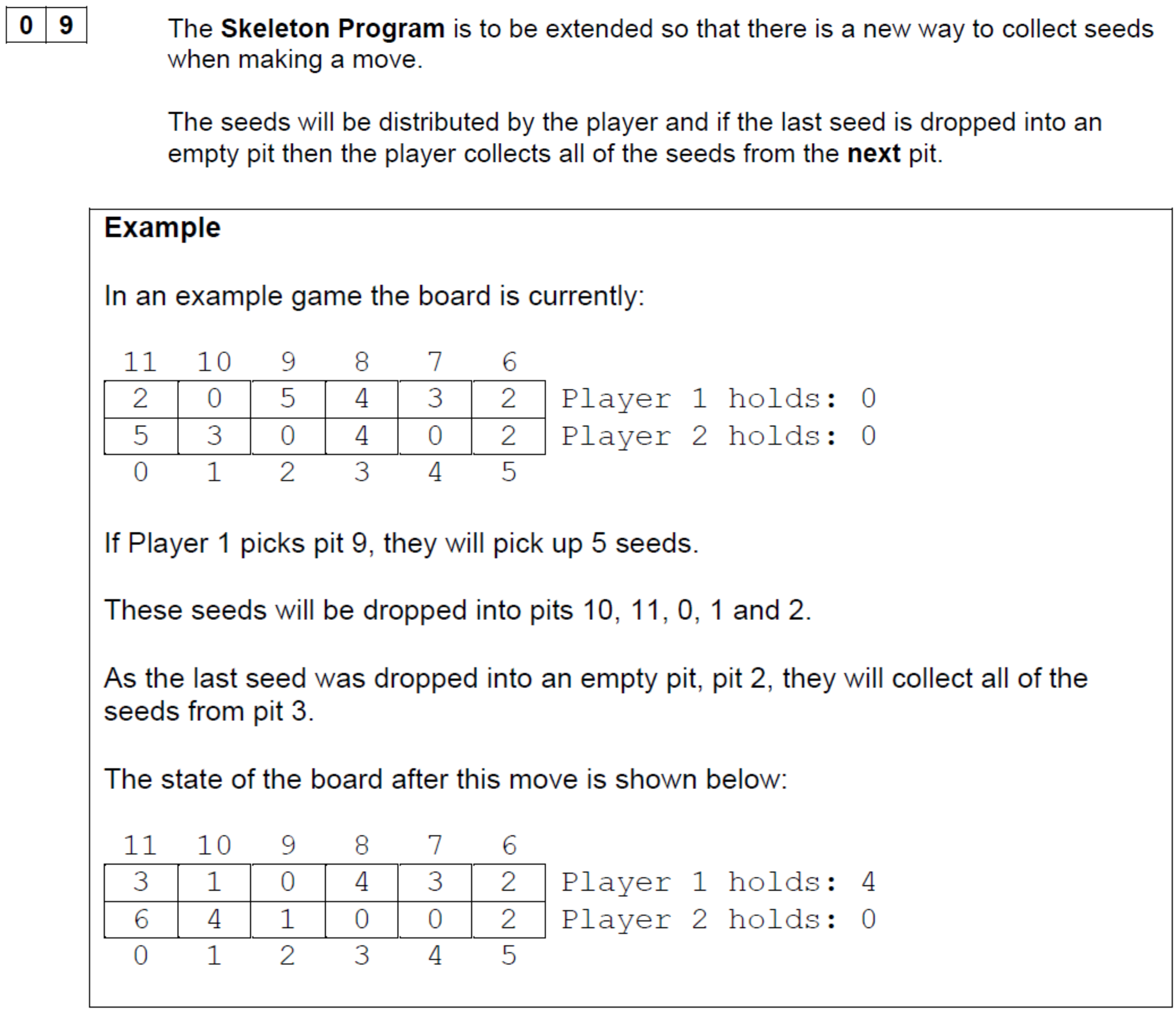
For part 8.3, mark points two, three and six from the mark scheme were awarded for displaying the message, calling CreateBoard (albeit with incorrect parameters) and generating a random number. The range for the random number was incorrect, the upper limit should have been PitCount-1, which prevented the award of the seventh mark point. As with part 8.2, the fact that the student did not copy all of the subroutine into the electronic answer document meant that mark point one could not be awarded as the location of the code could not be seen. Whilst the student has not produced a working solution, they have sensibly done what they could and so achieved some of the available marks.

For part 8.4, the mark was not awarded. The student had made a number of errors in parts 8.2 and 8.3 so was unable to provide testing evidence to show that their code worked.

Question number 9

Question number 9 introduced a new to collect seeds in the game. The Skeleton Program had to be modified so that if the last seed was dropped into an empty pit the player would collect all of the seeds from the next pit. The Section C programming questions are challenging and are intended to differentiate between the most able programmers, but all students should be encouraged to attempt them if they have time to, as there are usually some marks available for tackling the simpler parts of the task.

Question number part 9.1, 9.2 and 9.3



0 9 The **Skeleton Program** is to be extended so that there is a new way to collect seeds when making a move.

The seeds will be distributed by the player and if the last seed is dropped into an empty pit then the player collects all of the seeds from the **next** pit.

Example

In an example game the board is currently:

```
 11  10   9   8   7   6
| 2 | 0 | 5 | 4 | 3 | 2 |  Player 1 holds: 0
| 5 | 3 | 0 | 4 | 0 | 2 |  Player 2 holds: 0
  0   1   2   3   4   5
```

If Player 1 picks pit 9, they will pick up 5 seeds.

These seeds will be dropped into pits 10, 11, 0, 1 and 2.

As the last seed was dropped into an empty pit, pit 2, they will collect all of the seeds from pit 3.

The state of the board after this move is shown below:

```
 11  10   9   8   7   6
| 3 | 1 | 0 | 4 | 3 | 2 |  Player 1 holds: 4
| 6 | 4 | 1 | 0 | 0 | 2 |  Player 2 holds: 0
  0   1   2   3   4   5
```

Task One

Create a new subroutine `CollectNext` that:

- takes the parameters `Pit` and `Player`
- calls the `DropSeeds` subroutine to place the seeds into the pits
- checks whether the last seed was dropped into an empty pit and, if it was, collects all the seeds in the next pit
- displays the message `Collected Seeds` if any seeds were collected.

Task Two

Change the `PlayGame` subroutine so that after getting the move from the player:

- it asks, with an appropriate prompt, if the player wishes to use the original way of collecting seeds or the new way of collecting seeds
- it allows the user to enter `A` for the original way of collecting seeds or `B` for the new way of collecting seeds
- it checks the entered value and calls either `MakeMove` or `CollectNext`, as appropriate.

Task Three

Test your changes by running the **Skeleton Program** and then:

- select `Play the test board`
- enter a pit number of `0`
- enter a value for a collection method of `B`
- enter a pit number of `11`
- enter a value for a collection method of `B`
- enter a pit number of `3`
- enter a value for the collection method of `A`

Evidence that you need to provide

Include the following in your Electronic Answer Document.

0 9 . 1 Your PROGRAM SOURCE CODE for the new subroutine `CollectNext`.

[9 marks]

0 9 . 2 Your PROGRAM SOURCE CODE for the subroutine `PlayGame`.

[5 marks]

0 9 . 3 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Mark scheme

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>09</td><td>1</td><td>

1 mark: Creation of new subroutine named `CollectNext`
1 mark: Subroutine passed parameters `Pit` and `Player`
1 mark: Appropriate call to `DropSeeds`
1 mark: A check that `Board[LastPit]` is equal to 1
1 mark: Working out position of next pit by adding 1
1 mark: Including a correct usage of MOD to account for wrapping around board
1 mark: Correct updating of `State` variable
1 mark: Correct updating of `Board` variable
1 mark: Displays message `Collected Seeds` when seeds are collected.

MAX 8 if any errors

Python

```
def CollectNext(Pit, Player):
    global Board, State  # optional
    LastPit = DropSeeds(Pit)
    if Board[LastPit] == 1:
        NextPit = (LastPit + 1) % len(Board)
        State[Player] =  State[Player] + Board[NextPit]
        Board[NextPit] = 0
        print('Collected Seeds')
```

VB

```
Sub CollectNext(Pit As Integer, Player As Integer)
    Dim LastPit As Integer = DropSeeds(Pit)
    If Board(LastPit) = 1 Then
        NextPit = (LastPit + 1) Mod Board.Length
        State(Player) = State(Player) + Board(NextPit)
        Board(NextPit) = 0
        Console.WriteLine("Collected Seeds")
    End If
```

</td><td>9

AO3=9</td></tr>
</table>

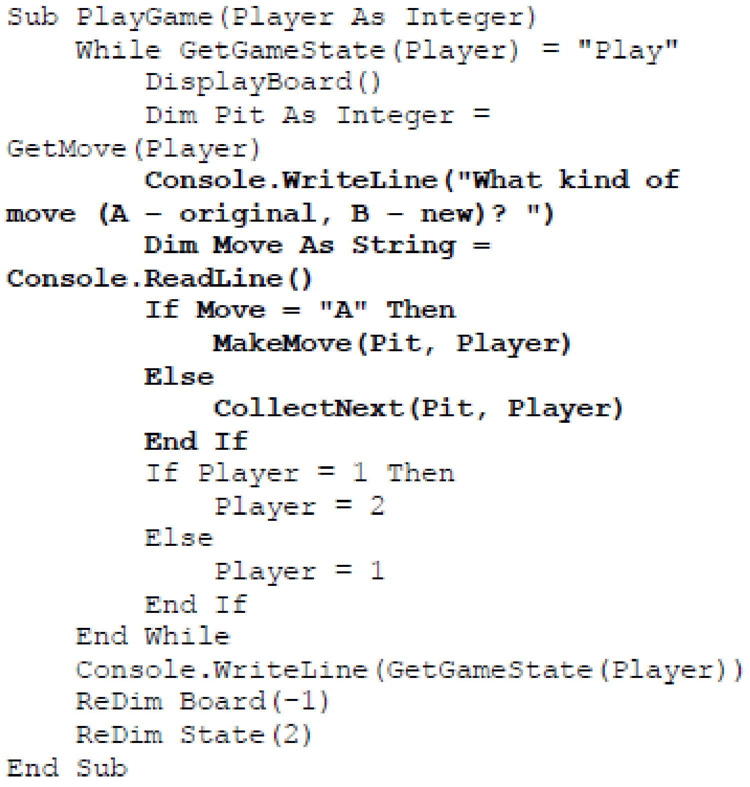
C#

```
private static void CollectNext(int Pit,
int Player)
{
    int LastPit = DropSeeds(Pit);
    if (Board[LastPit] == 1)
    {
        NextPit = (LastPit + 1) %
Board.Length;
        State[Player] = State[Player] +
Board[NextPit];
        Board[NextPit] = 0;
        Console.WriteLine("Collected
Seeds");
    }
}
```

Question	Part	Marking guidance	Total marks
09	2	**1 mark:** Appropriate prompt to ask for kind of move placed after use of `GetMove` **1 mark:** Selection statement setup to check response **1 mark:** Call to `MakeMove` when option A has been chosen **1 mark:** Call to `CollectNext` **1 mark:** Subroutines called under correct circumstances. **MAX 4 if any errors** **Python**	5 AO3=5

```
def PlayGame(Player):
    global Board, State
    while GetGameState(Player) == 'Play':
        DisplayBoard()
        Pit = GetMove(Player)
        Move = input('What kind of move (A - original, B - new)? ')
        if Move == 'A':
            MakeMove(Pit, Player)
        else:
            CollectNext(Pit, Player)
        if Player == 1:
            Player = 2
        else:
            Player = 1
    print(GetGameState(Player))
    Board = []
    State = []
```

VB



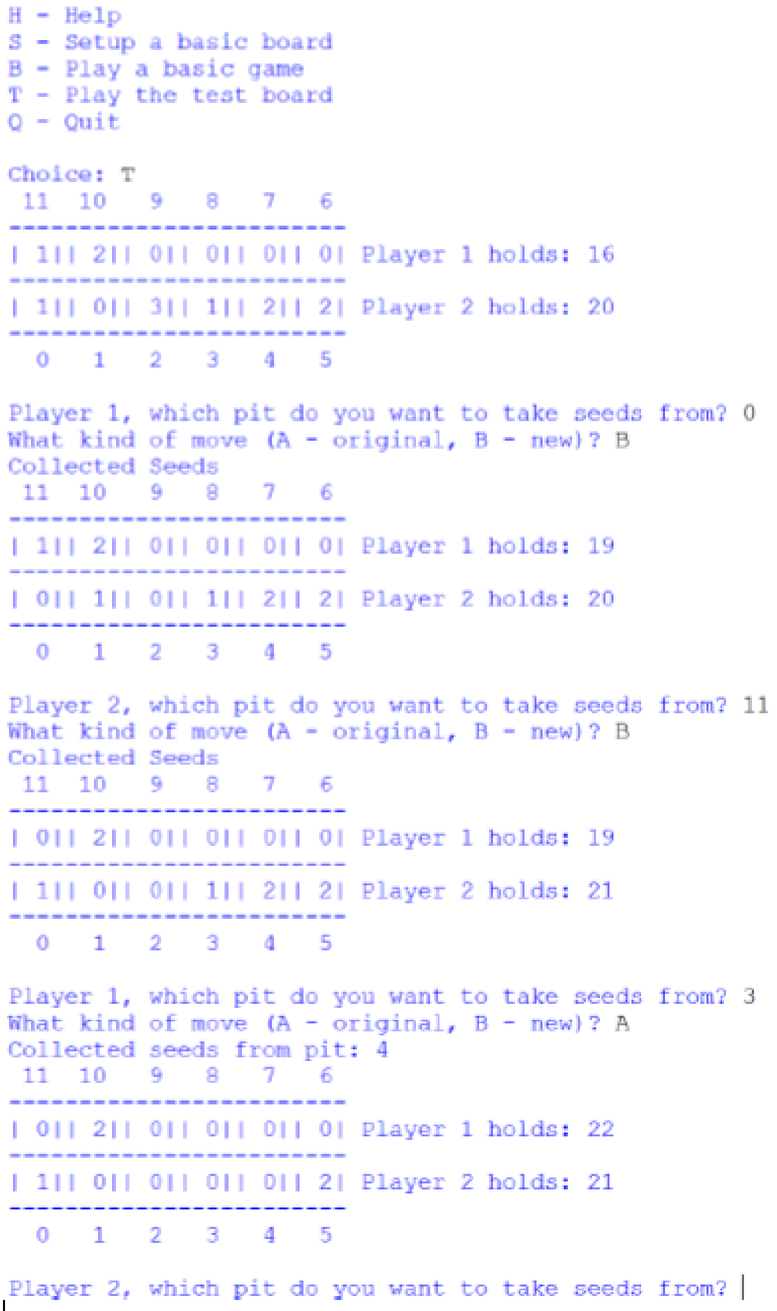
```
Sub PlayGame(Player As Integer)
    While GetGameState(Player) = "Play"
        DisplayBoard()
        Dim Pit As Integer =
GetMove(Player)
        Console.WriteLine("What kind of
move (A - original, B - new)? ")
        Dim Move As String =
Console.ReadLine()
        If Move = "A" Then
            MakeMove(Pit, Player)
        Else
            CollectNext(Pit, Player)
        End If
        If Player = 1 Then
            Player = 2
        Else
            Player = 1
        End If
    End While
    Console.WriteLine(GetGameState(Player))
    ReDim Board(-1)
    ReDim State(2)
End Sub
```

<table>
<tr>
<td></td>
<td></td>
<td>

C#

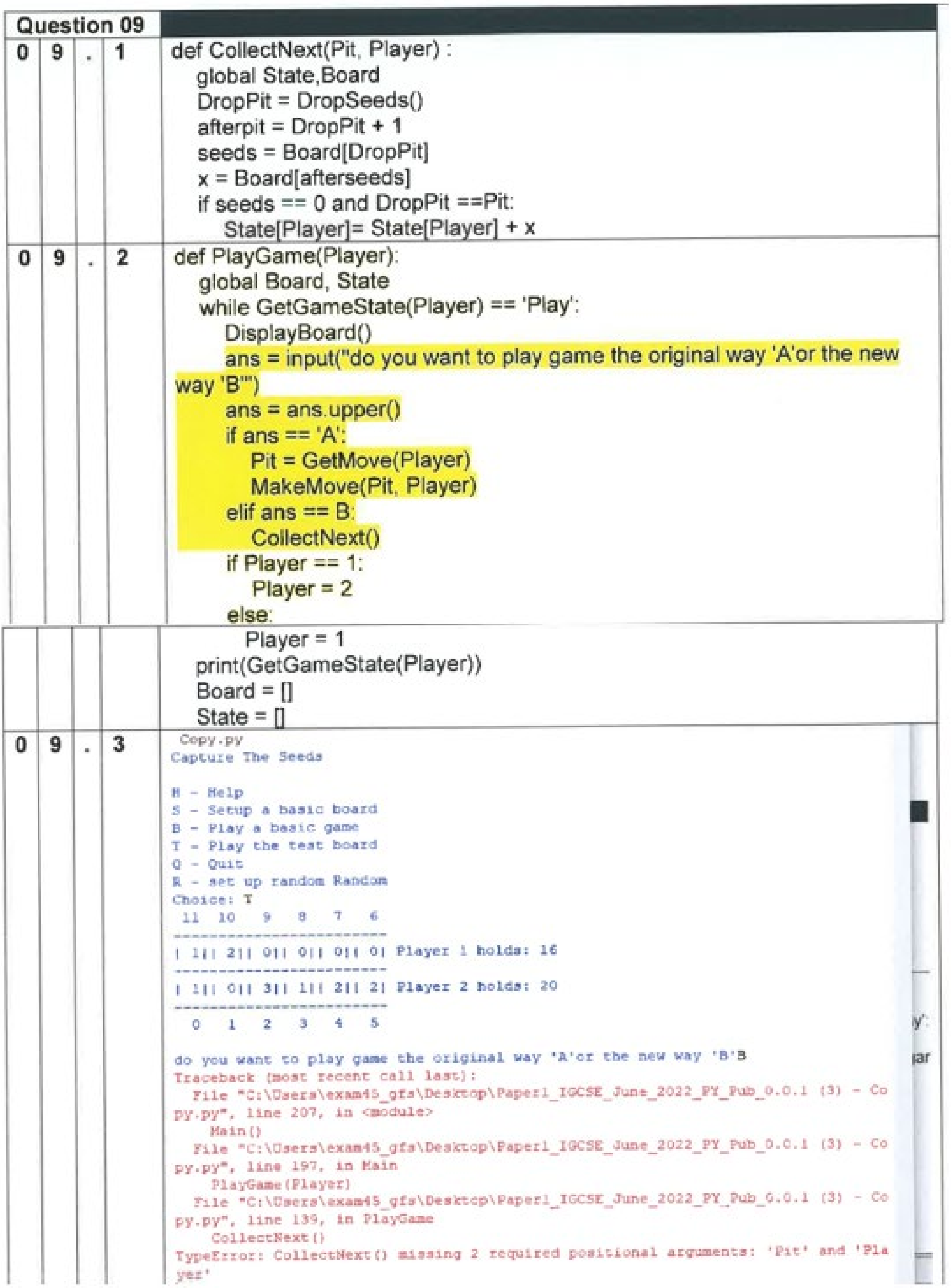
```
private static void PlayGame(int Player)
{
    while (GetGameState(Player) == "Play")
    {
        DisplayBoard();
        int Pit = GetMove(Player);
        Console.WriteLine("What kind of
move (A – original, B – new)? ");
        string Move = Console.ReadLine();
        if (Move == "A")
        {
            MakeMove(Pit, Player);
        }
        else
        {
            CollectNext(Pit, Player);
        }
        if (Player == 1)
            Player = 2;
        else
            Player = 1;
    }
Console.WriteLine(GetGameState(Player));
    Board = new int[0];
    State = new int[3];
}
```

</td>
<td></td>
</tr>
</table>

Question	Part	Marking guidance	Total marks
09	3	Screen capture(s) must match code from Q9.1 and Q9.2 **1 mark:** Screen capture shows the test being completed correctly. 	1 AO3=1

```
H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 0
What kind of move (A - original, B - new)? B
Collected Seeds
 11  10   9   8   7   6
------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 19
------------------------
| 0|| 1|| 0|| 1|| 2|| 2| Player 2 holds: 20
------------------------
  0   1   2   3   4   5

Player 2, which pit do you want to take seeds from? 11
What kind of move (A - original, B - new)? B
Collected Seeds
 11  10   9   8   7   6
------------------------
| 0|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 19
------------------------
| 1|| 0|| 0|| 1|| 2|| 2| Player 2 holds: 21
------------------------
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 3
What kind of move (A - original, B - new)? A
Collected seeds from pit: 4
 11  10   9   8   7   6
------------------------
| 0|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 22
------------------------
| 1|| 0|| 0|| 0|| 0|| 2| Player 2 holds: 21
------------------------
  0   1   2   3   4   5

Player 2, which pit do you want to take seeds from? |
```

Response A



Question 09

0 9 . 1

```
def CollectNext(Pit, Player) :
    global State,Board
    DropPit = DropSeeds()
    afterpit = DropPit + 1
    seeds = Board[DropPit]
    x = Board[afterseeds]
    if seeds == 0 and DropPit ==Pit:
        State[Player]= State[Player] + x
```

0 9 . 2

```
def PlayGame(Player):
    global Board, State
    while GetGameState(Player) == 'Play':
        DisplayBoard()
        ans = input("do you want to play game the original way 'A'or the new
way 'B'")
        ans = ans.upper()
        if ans == 'A':
            Pit = GetMove(Player)
            MakeMove(Pit, Player)
        elif ans == B:
            CollectNext()
        if Player == 1:
            Player = 2
        else:
            Player = 1
    print(GetGameState(Player))
    Board = []
    State = []
```

0 9 . 3

```
 Copy.py
Capture The Seeds

H - Help
S - Setup a basic board
B - Play a basic game
T - Play the test board
Q - Quit
R - set up random Random
Choice: T
 11  10   9   8   7   6
-------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
-------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
-------------------------
  0   1   2   3   4   5

do you want to play game the original way 'A'or the new way 'B'B
Traceback (most recent call last):
  File "C:\Users\exam45_gfs\Desktop\Paper1_IGCSE_June_2022_PY_Pub_0.0.1 (3) - Co
py.py", line 207, in <module>
    Main()
  File "C:\Users\exam45_gfs\Desktop\Paper1_IGCSE_June_2022_PY_Pub_0.0.1 (3) - Co
py.py", line 197, in Main
    PlayGame(Player)
  File "C:\Users\exam45_gfs\Desktop\Paper1_IGCSE_June_2022_PY_Pub_0.0.1 (3) - Co
py.py", line 139, in PlayGame
    CollectNext()
TypeError: CollectNext() missing 2 required positional arguments: 'Pit' and 'Pla
yer'
```

Commentary

This response received **3 marks** for part **9.1**, **2 marks** for part **9.2** and **0 marks** for part **9.3**.

For part 9.1, the first, second and fifth mark points from the mark scheme were awarded. The first two mark points were awarded because the student created a subroutine with the correct name and parameters – all students should be encouraged to attempt to do this, even if they are unable to tackle the rest of the question. The third mark point was not awarded as although there is a call to `DropSeeds`, this does not include the required parameter so it was not an appropriate call, which the mark scheme required. The fifth mark point was awarded as the student does add one onto `DropPit` to calculate the position of the next pit.

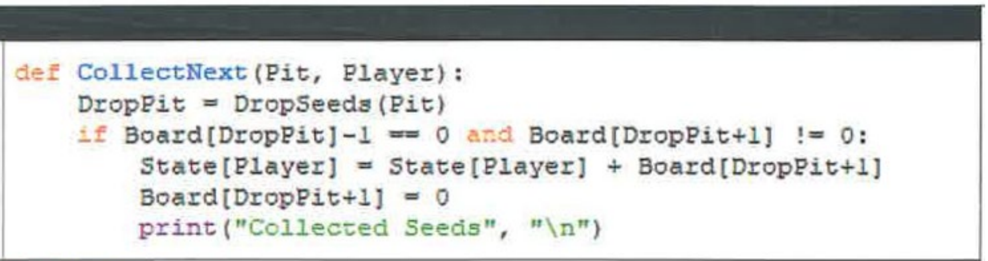
For part 9.2, mark points two and three from the mark scheme were awarded. There is a selection statement to check for the type of response and `MakeMove` is called for the original type of move. The missing quotation marks around the B were ignored for mark point two, as this only required that there was a selection statement, not that it fully worked. Mark point one was not awarded as the prompt was not displayed after `GetMove` was called, mark point three was not awarded as the call to `CollectNext` would not work because it has no parameters and mark point five was not awarded as the circumstances under which the subroutines were called was not correct owing to the missing quotation marks around the `B`.

For part 9.3, no marks were awarded. It can be seen from the test evidence (as well as the code) that the solution did not work.

Response B

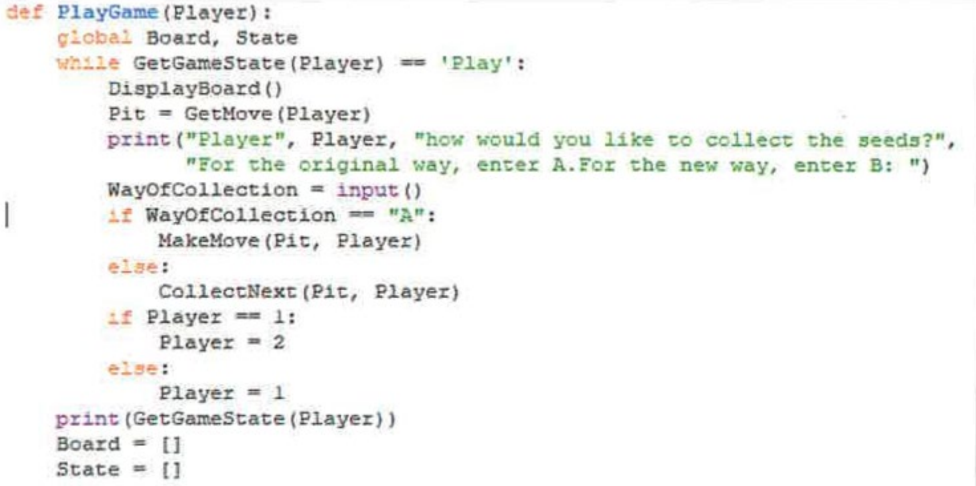
Question 09

0 9 . 1



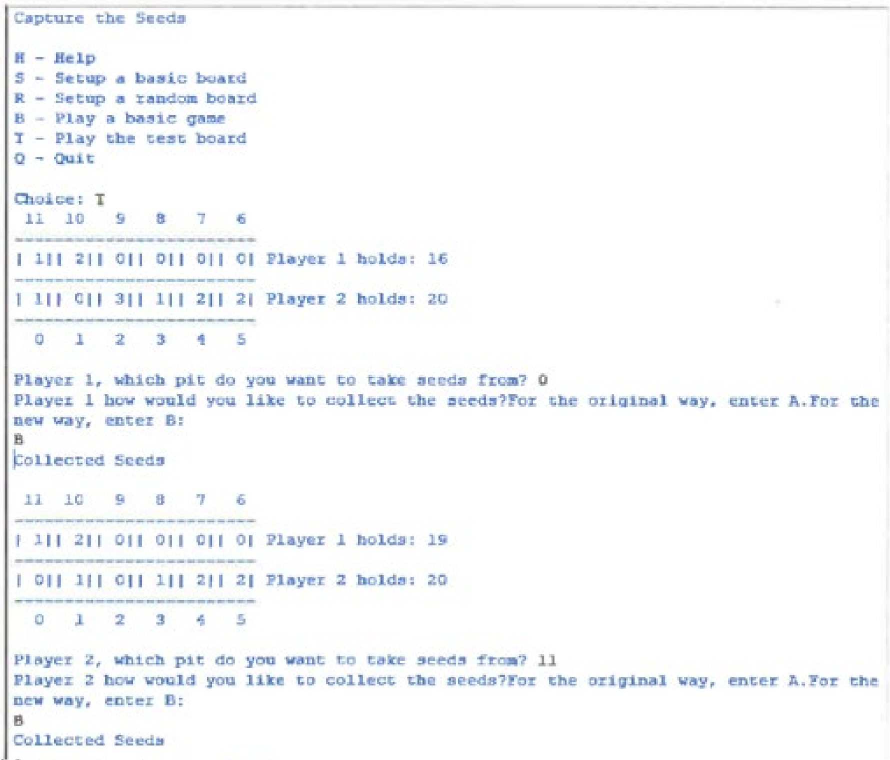
```
def CollectNext(Pit, Player):
    DropPit = DropSeeds(Pit)
    if Board[DropPit]-1 == 0 and Board[DropPit+1] != 0:
        State[Player] = State[Player] + Board[DropPit+1]
        Board[DropPit+1] = 0
        print("Collected Seeds", "\n")
```

0 9 . 2

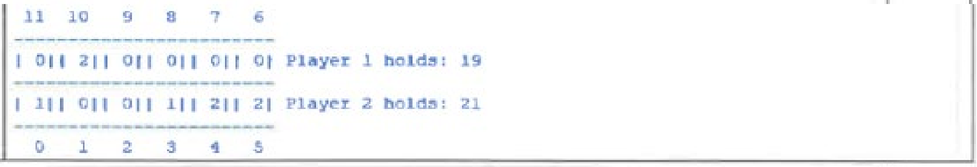


```
def PlayGame(Player):
    global Board, State
    while GetGameState(Player) == 'Play':
        DisplayBoard()
        Pit = GetMove(Player)
        print("Player", Player, "how would you like to collect the seeds?",
              "For the original way, enter A.For the new way, enter B: ")
        WayOfCollection = input()
        if WayOfCollection == "A":
            MakeMove(Pit, Player)
        else:
            CollectNext(Pit, Player)
        if Player == 1:
            Player = 2
        else:
            Player = 1
    print(GetGameState(Player))
    Board = []
    State = []
```

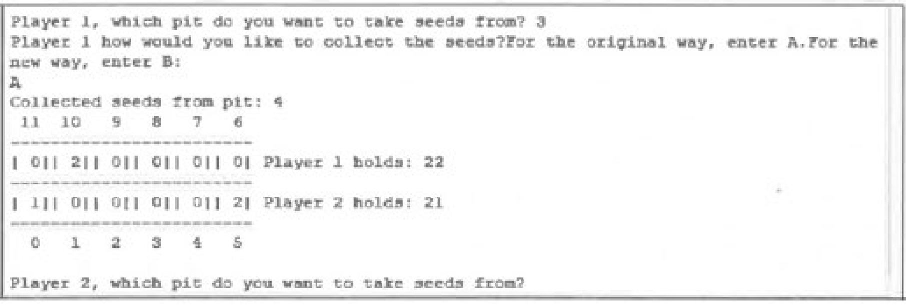
09.3



```
Capture the Seeds

H - Help
S - Setup a basic board
R - Setup a random board
B - Play a basic game
T - Play the test board
Q - Quit

Choice: T
 11  10   9   8   7   6
-------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 16
-------------------------
| 1|| 0|| 3|| 1|| 2|| 2| Player 2 holds: 20
-------------------------
  0   1   2   3   4   5

Player 1, which pit do you want to take seeds from? 0
Player 1 how would you like to collect the seeds?For the original way, enter A.For the
new way, enter B:
B
Collected Seeds

 11  10   9   8   7   6
-------------------------
| 1|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 19
-------------------------
| 0|| 1|| 0|| 1|| 2|| 2| Player 2 holds: 20
-------------------------
  0   1   2   3   4   5

Player 2, which pit do you want to take seeds from? 11
Player 2 how would you like to collect the seeds?For the original way, enter A.For the
new way, enter B:
B
Collected Seeds
```



```
 11  10   9   8   7   6
-------------------------
| 0|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 19
-------------------------
| 1|| 0|| 0|| 1|| 2|| 2| Player 2 holds: 21
-------------------------
  0   1   2   3   4   5
```



```
Player 1, which pit do you want to take seeds from? 3
Player 1 how would you like to collect the seeds?For the original way, enter A.For the
new way, enter B:
A
Collected seeds from pit: 4
 11  10   9   8   7   6
-------------------------
| 0|| 2|| 0|| 0|| 0|| 0| Player 1 holds: 22
-------------------------
| 1|| 0|| 0|| 0|| 0|| 2| Player 2 holds: 21
-------------------------
  0   1   2   3   4   5

Player 2, which pit do you want to take seeds from?
```

Commentary

This was an excellent response received **7 marks** for part **9.1**, **5 marks** for part **9.2** and **1 mark** for part **9.3**.

For part 9.1, the two mark points that were not awarded were mark points four and six on the mark scheme

Mark point six was not awarded as the student did not take account of needing to wrap around the board after adding 1 onto the pit number, for example by using the MOD operator.

For mark point four, the student had to check that `Board[LastPit]` was equal to 1, ie that the last pit used was empty before the seed was dropped into it. In their solution, the student has checked that `Board[DropPit]-1` is equal to 0, which achieves the same thing. If their `if` statement had only included this one condition then mark point four would have been awarded, but the second condition `Board[DropPint+1] != 0` would stop this working under some circumstances, so this mark was not awarded.

For parts 9.2 and 9.3, full marks were awarded. Whilst the code would not work under all circumstances, it did work using the test data given in question part 9.3, so this mark was awarded.