OXFORD AQA
INTERNATIONAL QUALIFICATIONS

The international exam board *that puts fairness first*

OxfordAQA International AS/A-Level

Computer Science (9645)

CS03 Example responses (Additional specimen)

For teaching from September 2025 onwards

For International GCSE exams in May/June 2026 onwards

Contents

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

Introduction

This guide includes example responses to questions from CS03 of the second set of Specimen Assessment Materials for International A-Level Computer Science (9645).

The non-multiple choice questions are presented with example responses and commentary from a senior examiner.

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.

Example Responses

Question 1

0 1 **Figure 1** shows three one dimensional arrays called A, B and C.

Figure 1

	A	**B**	**C**
[0]	*	1	2
[1]	+	3	4
[2]	9	-1	-1
[3]	5	-1	-1
[4]	3	-1	-1

0 1 . 1 Describe how the arrays shown in **Figure 1** could be used to create a tree data structure.

[3 marks]

Question part 1.1

Question 1.1 tests Assessment Objective AO3 and requires students to describe how the arrays could be used to form a Tree structure.

Mark Scheme

Question	Part	Marking guidance	Total marks
01	**1**	Array A stores the contents of each node; Array B stores the index/location for the left-pointer of the node; Array C stores the index for/location the right-pointer of the node; **R** Array B/C stores the left/right node **Must be clear that Array B and Array C refer to a location/index**	**3** **AO3 = 3**

Response A

The three arrays together can be used to draw a tree. Array A has the contents of each node. Array B has the location ID for the node to the left of the current node, and Array C has the location ID for the node to the right of the current node.

Commentary

This response received **3 marks**. The student had clearly identified the purpose of each array and had made clear that Arrays B and C refer to locations (location ID is acceptable).

Response B

Array A is the node. Array B is the node on the left and Array C is the node on the right.

Commentary

This response received **0 marks**. Although they understood the general purpose of each array, they were missing technical details about each array which were needed. It must be clear that Array A is referring to the contents / data / value of the node (saying the node is not enough), and Array B and C refer to pointers/locations/indices for the left and right nodes (not the nodes themselves).

Question part 1.2

Question 1.2 tests Assessment Objective AO3 and requires students to

0 1 . 2 State **one** property that a graph can have but a tree cannot have.

[1 mark]

Mark Scheme

Question	Part	Marking guidance	Total marks
01	2	Tree cannot contain a cycle // Tree cannot contain a closed path where all intermediate nodes are independent; Tree must be connected // Graph can be disconnected; **A** Graphs do not need a root node	1 AO3 = 1

Response A

A Graph can have a cycle, but a Tree cannot have any cycles

Commentary

This response received **1 mark.** The student had identified the key difference between a graph and a tree

Response B

A Tree doesn't contain loops.

Commentary

This response received **0 marks**. A loop is not the same as a cycle, although they are often seen as the same thing. A loop refers to an iterative structure, and a cycle is a technical term in graph theory which refers to a closed path with unique intermediate nodes. Students should be careful to use the correct key terms for the question.

Question part 1.3

Question 1.3 tests Assessment Objective AO3 and requires students to trace an algorithm described in **Figure 2**.

0 1 . 3 The pseudocode shown in **Figure 2** uses the three arrays (A, B and C) from **Figure 1** to produce an output.

Figure 2

```
SUBROUTINE traverse(pos)
     IF b[pos] > 0 THEN
          traverse(b[pos])
     ENDIF
     IF c[pos] > 0 THEN
          traverse(c[pos])
     ENDIF
     OUTPUT a[pos]
ENDSUBROUTINE
```

Complete **Table 1** by tracing the pseudocode in **Figure 2** for the subroutine call `Traverse(0)`

You should use the values from **Figure 1**.

You can add or remove rows in **Table 1** as needed.

Table 1

pos	OUTPUT

[4 marks]

Mark Scheme

Question	Part	Marking guidance	Total marks
01	3		4 **AO3 = 4**

Pos	Output
0	
1	
3	5
1	
4	3
1	+
0	
2	9
0	*

One mark for each box (as shown above):

- Traverse(1) will be made from Traverse(0)
- Traverse(3) will be made from Traverse(1) and it will output the number 5
- After returning to Traverse(1), a further call to Traverse(4) is made with output 3, and then outputting + for Traverse(1) and returning to the original call Traverse(0)
- Traverse(0) will make a further call to Traverse(2) and output 9, before returning to Traverse(0) to output *

Alternate table:

`pos`	`OUTPUT`
0	
1	
3	5
4	3
1	+
2	9
0	*

Response A

Pos	Output
1	
3	5
1	
4	3
1	+
2	9
0	*

Commentary

This response received **3 marks**. The student correctly traced the algorithm but had forgotten to include the initial value for Pos when called (0). This student had received marks for MP2, MP3, and MP4.

Response B

Pos	Output
0	
1	
3	5

Commentary

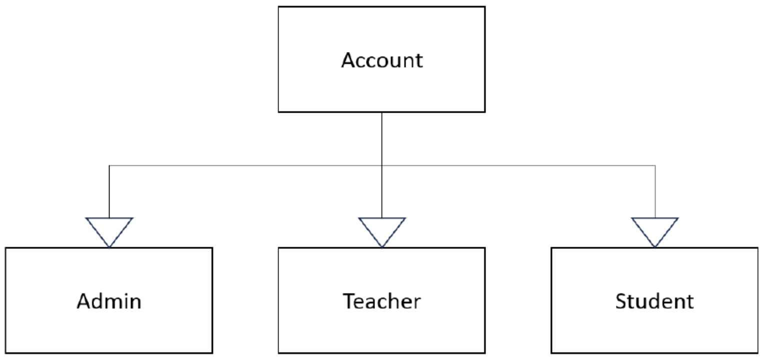
This response received **2 marks**. The student had identified that recursive calls are made when Pos is 1 and 3 but had ended the algorithm early after the first output. They had been awarded only MP1 and MP2 (using the alternate table in the mark scheme).

Question 2

0 2 A programmer is writing a program which can create different types of accounts for a school network. The school have asked for three different types of account: **Teacher, Student,** and **Admin**.

The programmer has decided to plan their solution with a class diagram. The class diagram is shown in **Figure 3.**

Figure 3



Question 2 part 2.1

Question 2.1 tests Assessment Objective AO3 and requires students to identify the mistake.

0 2 . 1 Describe the mistake that the programmer has made in **Figure 3**.

[1 mark]

Mark Scheme

Question	Part	Marking guidance	Total marks
02	1	The arrows should point towards Account // The arrows should point up; **NE** arrows	1 AO3 = 1

Response A

The inheritance arrows are the wrong way, they should point to account instead.

Commentary

This response received **1 mark.** The student had identified the mistake and made it clear where the arrows should be pointing.

Response B

Account should be under Admin, Teacher, and Student

Commentary

This response received **0 marks.** The student had not identified the mistake. Even if the Account box moved under the others, the arrows would indicate an incorrect relationship.

Question 2 part 2.3

Question 2.3 tests Assessment Objective AO3 and requires students to describe the differences between private and protected attributes.

0 2 . 3 Describe the difference between an attribute that has a private access modifier and an attribute that has a protected access modifier.

[2 marks]

Mark Scheme

Question	Part	Marking guidance	Total marks
02	**3**	Private means it can only be accessed / seen in the class it is in; Protected means it can be accessed / seen in the class it is in and in any subclasses // Protected means it can be accessed / seen in the class it is in and in any classes derived / inheriting from it; **A** Protected is like private but it can also be accessed in classes derived / inheriting from it; **R** Protected can be accessed by any class	**2** **AO3 = 2**

Response A

Private attributes can only be accessed inside the class, but protected attributes can also be accessed by any subclasses.

Commentary

This response received **2 marks**. The student had explained private attributes and also explained what protected attributes do in addition to private attributes. If the student did not include the word *also* in their response, then they would not get the second mark in this example.

Response B

Protected attributes can be accessed by any class. Private attributes can only be accessed by one class.

Commentary

This response received **0 marks**. The student stated that protected attributes are accessed by any class but it is only true for the class it is defined in and any of its subclasses (not all classes). Also, their second point is not clear enough. They needed to state that the private attribute can only be accessed by the class it is defined in. This would make clear that it can't be declared in one class and accessible by another class.

Question 3

Question 3.1 tests Assessment Objective AO3 and requires students to describe collisions and explain how rehashing can reduce the number of collisions.

0 3 A programmer is trying to create a hash table to store their data. One issue they have is that there are many collisions in their hash table.

0 3 . 1 Describe what is meant by the term collision in a hash table and how the programmer could use rehashing to reduce the number of collisions.

[3 marks]

Mark Scheme

Question	Part	Marking guidance	Total marks
03	1	**Maximum one mark for description of collision:** A collision is when two items of data hash to the same hash value; **A** by example (so long as example is clear) **Maximum two marks for explanation of rehashing reducing collisions:** Rehashing will create a bigger table allowing for more records to be stored in unique locations; A new hashing algorithm is created to reduce the chance of future collisions; All data from the previous hash table is passed through the new hashing algorithm to be given a new location in the bigger hash table;	3 AO3 = 3

Response A

A collision occurs when two different inputs create the same output when passed into a hashing algorithm. To reduce this, you could use rehashing which means you create a bigger table and put all data from the old table into the new table using a new hashing algorithm (which will tell it where to go).

Commentary

This response received **3 marks**. The student identified a collision and had used appropriate key terms to explain how it occurs. They had also explained how to re-hash the table of data, making clear that a new, larger table is created and the data in the original table is passed through a new hashing algorithm.

Response B

A collision is when two inputs hash to the same value, and would have to go in the same location in the table. This is not possible, so a new table would have to be created and all of the data would be stored in the new table.

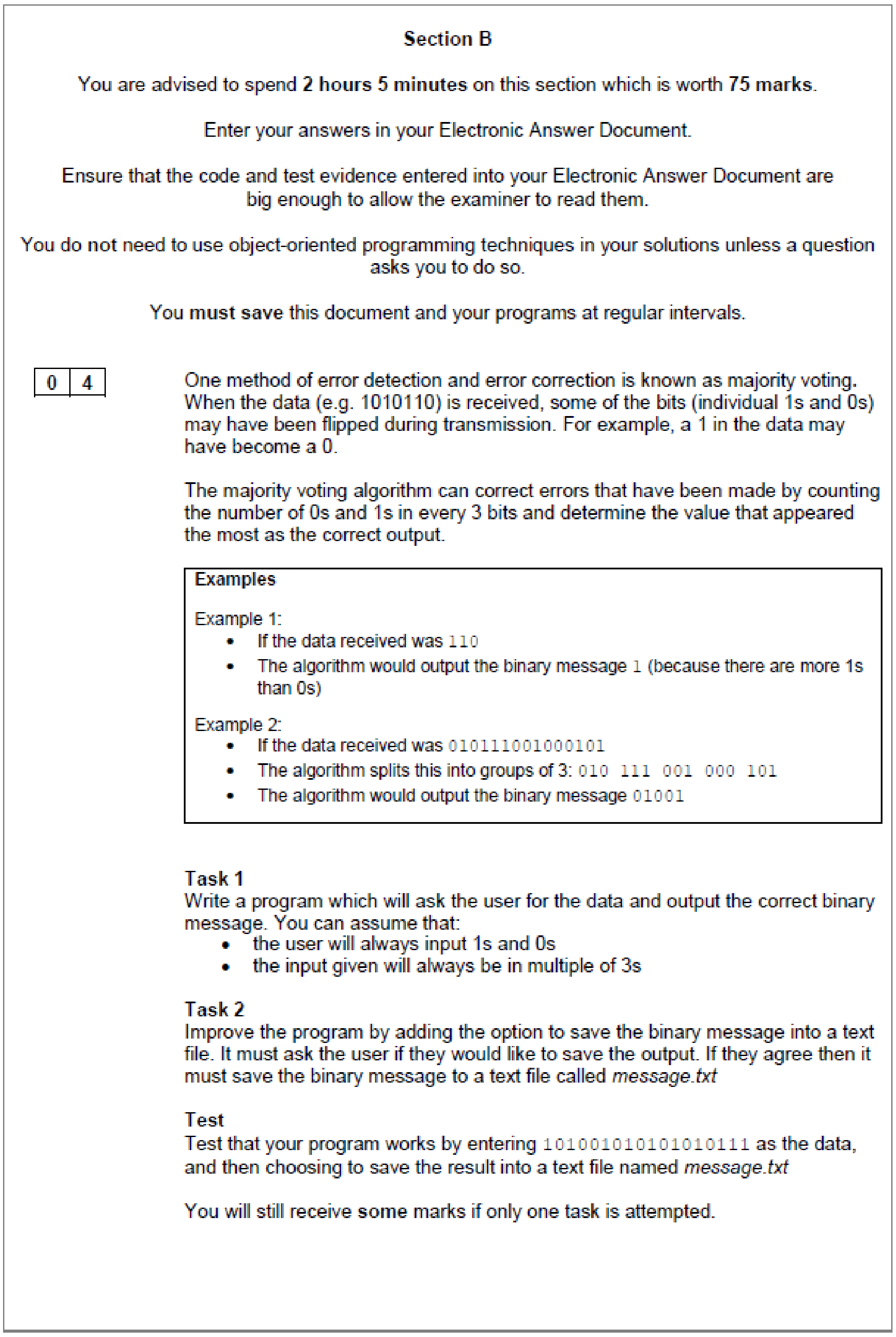
Commentary

This response received **1 mark**. The student had correctly described a collision using key terminology. However, when describing re-hashing they should make it clear that the new table is bigger/larger (otherwise you would still get the same issue). They also did not make clear that the hashing algorithm would have to change.

Question 4

Question 4 tests Assessment Objective AO3 and requires students to write a program based on the majority voting system. Knowledge of majority voting is not tested directly, however, the student's ability to provide a coded solution that meets the task instructions is tested.

Question 4 part 4.1 and 4.2



Section B

You are advised to spend **2 hours 5 minutes** on this section which is worth **75 marks**.

Enter your answers in your Electronic Answer Document.

Ensure that the code and test evidence entered into your Electronic Answer Document are big enough to allow the examiner to read them.

You do not need to use object-oriented programming techniques in your solutions unless a question asks you to do so.

You **must save** this document and your programs at regular intervals.

0 4 One method of error detection and error correction is known as majority voting. When the data (e.g. 1010110) is received, some of the bits (individual 1s and 0s) may have been flipped during transmission. For example, a 1 in the data may have become a 0.

The majority voting algorithm can correct errors that have been made by counting the number of 0s and 1s in every 3 bits and determine the value that appeared the most as the correct output.

Examples

Example 1:

- If the data received was `110`
- The algorithm would output the binary message `1` (because there are more 1s than 0s)

Example 2:

- If the data received was `010111001000101`
- The algorithm splits this into groups of 3: `010 111 001 000 101`
- The algorithm would output the binary message `01001`

Task 1
Write a program which will ask the user for the data and output the correct binary message. You can assume that:

- the user will always input 1s and 0s
- the input given will always be in multiple of 3s

Task 2
Improve the program by adding the option to save the binary message into a text file. It must ask the user if they would like to save the output. If they agree then it must save the binary message to a text file called *message.txt*

Test
Test that your program works by entering `101001010101010111` as the data, and then choosing to save the result into a text file named *message.txt*

You will still receive **some** marks if only one task is attempted.

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

0 4 . 1 Your PROGRAM SOURCE CODE for your entire program.

[11 marks]

0 4 . 2 The output produced when you carry out the test.

[2 marks]

Mark scheme

Question	Part	Marking guidance	Total marks

04	1	**Task 1 requirements**
		One mark each for:
		• Asking for an input from the user and storing the response in a variable
		• Creating a loop
		• Iterating an appropriate number of times (e.g. the length of the variable // for every character in the String)
		• Successfully counting the number of ones // number of zeroes in the first 3 characters
		• Outputting a one or a zero based on which had the majority from the first 3 characters
		• Repeating this process for every set of 3 characters in the string
		• The final correct output being on a single line (for readability)
		
		Task 2 requirements
		One mark each for:
		• Giving the user the choice to save to a file and storing their response in a variable
		• Correctly opening the connection to the text file message.txt
		• Correctly writing the correct output into the text file message.txt if the user said yes
		• Correctly closing the connection to the text file message.txt
		
		<u>Exemplar Solutions</u>
		
		Python
		`payload = input("Please enter the data:")`
		`count = 0`
		`numOfOnes = 0`
		`finalOutput = ""`
		`for character in payload:`
		`    count += 1`
		`    if character == "1":`
		`        numOfOnes += 1`
		`    if count == 3:`
		`        if numOfOnes >= 2:`
		`            finalOutput += "1"`
		`        else:`
		`            finalOutput += "0"`
		`        count = 0`
		`        numOfOnes = 0`
		`print("Final output is",finalOutput)`
		`choice = input("Would you like to save the data?")`
		`if choice == "yes":`
		`    file = open("message.txt","w")`
		`    file.write(finalOutput)`
		`    file.close()`
			11
			
			AO3 = 11

C#

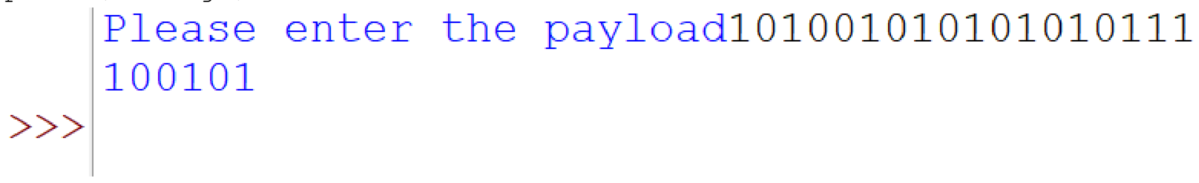
```
string payload = "";
int count = 0;
int numOfOnes = 0;
string finalOutput = "";

Console.WriteLine("Please enter the data:");
payload = Console.ReadLine();
foreach (Char character in payload)
{
    count++;
    if (character.ToString() == "1")
    {
        numOfOnes++;
    }
    if (count == 3)
    {
        if(numOfOnes >= 2)
        {
            finalOutput += "1";
        }
        else
        {
            finalOutput += "0";
        }
        count = 0;
        numOfOnes = 0;
    }
}
Console.WriteLine("Final output is " +
finalOutput);
string choice = "";
Console.WriteLine("Would you like to save the
data?");
choice = Console.ReadLine();
if (choice == "yes")
{
    StreamWriter fileWriter = new
StreamWriter("message.txt");
    fileWriter.Write(finalOutput);
    fileWriter.Close();
}
```

VB.Net

```
Module Program
    Sub Main(args As String())
        Dim payload As String
        Dim count As Integer = 0
        Dim numOfOnes As Integer = 0
        Dim finalOutput As String = ""
        Console.WriteLine("Please enter your data")
        payload = Console.ReadLine
        For Each character In payload
            count += 1
            If character = "1" Then
                numOfOnes += 1
            End If
            If count = 3 Then
                If numOfOnes >= 2 Then
                    finalOutput += "1"
                Else
                    finalOutput += "0"
                End If
                count = 0
                numOfOnes = 0
            End If
        Next
        Console.WriteLine("Final output is..." +
finalOutput)
        Dim choice As String
        Console.WriteLine("Would you like to save
the data?")
        choice = Console.ReadLine
        If choice = "yes" Then
          Dim fileWriter As New
IO.StreamWriter("message.txt")
            fileWriter.Write(finalOutput)
            fileWriter.Close()
        End If
    End Sub
End Module
```

Response A

```
payload = input("Please enter the payload")

tick = 0

value = 0

message = ""

for i in range(len(payload)):

    tick += 1

    if payload[i] == "0":

        value -= 1

    else:

        value += 1

    if tick == 3:

        if value < 0:

            message += "0"

        else:

            message += "1"

        tick = 0

        value = 0

print(message)
```



```
     Please enter the payload101001010101010111
     100101
>>>
```

Commentary

This response received **7 marks** (MP1 to MP7) for the code and **1 mark** for the screenshot.

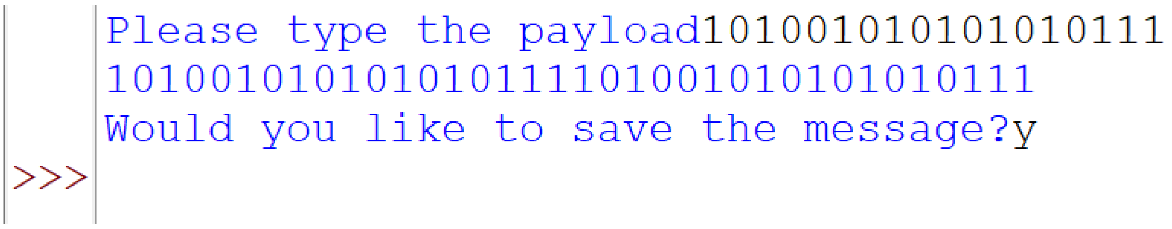
This was a part response to the question. The student had completed all of task one by asking for an input and storing in a suitable variable (payload). They also created a loop which iterates through for the length of payload. They also correctly counted the number of one/zeroes using a variable which went up or down (value). Also, after the first 3 digits have been read the algorithms correctly reset the variables for tick and value. The evidence also proved that the algorithm works correctly for intended inputs.

However, no attempt was made at saving the output to a text file. It's important to note that simply asking the user for the option and storing their response is enough to get an extra mark in this question. Code with sets up a task can usually be awarded some marks even if the key part of the task was not attempted.

The screenshot evidence showed the correct output from task one, but no output for task two and so this was awarded **1 mark**.

Response B

```
message1 = input("Please type the payload")

for character in message1:

    if character == "0":

        count = 0

    else:

        count = 1

    message1 = message1 + str(count)

print(message1)

save = input("Would you like to save the message?")

if save == "y":

    f = open("message.txt","w")

    f.write(message1)

    f.close()
```



```
Please type the payload101001010101010111
101001010101010111101001010101010111
Would you like to save the message?y
>>>
```

Commentary

This response received **7 marks** (MP1 to MP3 and MP8, MP9, and MP11) for the code and **0 marks** for the screenshot.

This was also a part response to the question. The student had attempted both tasks but had not correctly completed the majority voting task.

The student had asked for an input and stored the response (MP1) in a suitable variable (message1). They also created a loop which iterates for every character (MP2 and MP3) in the string. However, inside the loop the code simply added the character that it has seen. An attempt has been made to keep track of how many (this is why count exists), however the student has not used it correctly. This meant that MP4 to MP7 could not be awarded as it did not look at every set of 3 characters, and it did not repeat that process for each set of 3.

The student completed task two correctly. There was a choice to save (MP8), and a connection to the text file (MP9) as well as the connection being closed at the end (MP11). However, MP10 requires the correct output to be stored into the text file and so this mark was not awarded because of the mistakes in task one. This meant that **7 marks** were awarded for the program code.

The screenshot evidence showed that the correct output was not given, and so was awarded **0 marks**.

Question 5

Question 5 tests Assessment Objective AO3 and requires students to create a Tree structure and perform a Pre-order tree traversal algorithm on the structure.

Question 5 part 5.1 and 5.2

0 5

A Node can be represented as a Class, which has these properties:

- Label: A string which stores the data of the node. This is required when a node is first instantiated. This property has a get method.
- LeftPtr: The node which is the left-child of the current node. This is first given a null value when instantiated. This property has get and set methods.
- RightPtr: The node which is the right-child of the current node. This is also given a null value when instantiated. This property also has get and set methods.

A binary tree is a tree where each node has at most two child nodes. One algorithm that can be executed on a binary tree is the pre-order traversal.

Figure 4 shows an example of a binary tree:

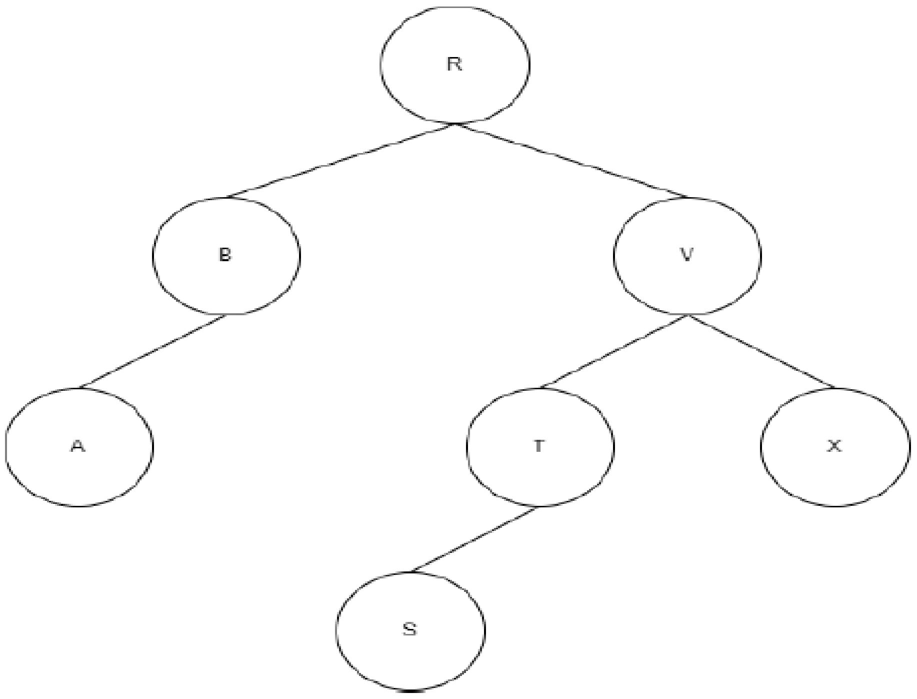


Figure 5 shows a recursive method to perform the pre-order tree traversal of a binary tree.

Figure 5

```
SUBROUTINE preOrder(currentNode)
      OUTPUT(Label of currentNode)
      IF currentNode has leftSubtree THEN
            preOrder(leftSubtree of currentNode)
      ENDIF
      IF currentNode has rightSubtree THEN
            preOrder(rightSubtree of currentNode)
      ENDIF
ENDSUBROUTINE
```

Task 1
Write a program that represents the binary tree in **Figure 5**. You must use an object-oriented approach for the nodes.

If you are working in Python, you should follow the convention of using two underscores __ to represent a private attribute.

Task 2
Write the `preOrder` method so that it performs a pre-order traversal of the binary tree.

To achieve full marks for this task, your program must use the recursive method outlined in **Figure 5** to perform the traversal.

Test

Test your program works by showing the program's output when a pre-order traversal is performed on the binary tree in **Figure 4**

Evidence that you need to provide

0 5 . 1 Your PROGRAM SOURCE CODE for your entire program.

[12 marks]

0 5 . 2 The output produced when you carry out the test.

[1 mark]

Mark Scheme

Question	Part	Marking guidance	Total marks

05	1	**Maximum 11 marks** if not fully working.
		
		1. Attempt at creating a class for node.
		2. Statement which correctly assigns a label to a node object.
		3. Correct representation of labels and pointers for each node (matching the given tree).
		4. Subroutine created for Pre-Order, including a parameter for current node.
		5. The subroutine calls itself.
		6. Subroutine will output current node.
		7. Output is at the start of the subroutine.
		8. Checks if current node has left child.
		9. Makes recursive call for the left child node if condition is met.
		10. Checks if current node has right child <u>after</u> checking left child.
		11. Makes recursive call for the right child node if condition is met.
		12. Correct result of pre-order traversal displayed.
		
		Note: the program must make a check for the left subtree before the right subtree. If right subtree is checked first, award points 8 and 9 and skip points 10 and 11.
		
		<u>**Example Solutions:**</u>
		Python
		`class Node:`
		`    def __init__(self, data):`
		`        self.__label = data`
		`        self.__LeftPtr = None`
		`        self.__RightPtr = None`
		`    def getLabel(self):`
		`        return self.__label`
		`    def getLeftPointer(self):`
		`        return self.__LeftPtr`
		`    def getRightPointer(self):`
		`        return self.__RightPtr`
		`    def setLeftPointer(self, node):`
		`        self.__LeftPtr = node`
		`    def setRightPointer(self, node):`
		`        self.__RightPtr = node`
		
		`def preOrder(currentNode):`
		`    print(currentNode.getLabel())`
		`    if currentNode.getLeftPointer() != None:`
		`        preOrder(currentNode.getLeftPointer())`
		`    if currentNode.getRightPointer() != None:`
		`        preOrder(currentNode.getRightPointer())`
		
		`R = Node("R")`
		`B = Node("B")`
		`V = Node("V")`
		`A = Node("A")`
			12
			
			AO3 = 12

```
T = Node("T")
X = Node("X")
S = Node("S")
R.setLeftPointer(B)
R.setRightPointer(V)
B.setLeftPointer(A)
V.setLeftPointer(T)
V.setRightPointer(X)
T.setLeftPointer(S)
preOrder(R)
```

C#

```
Node R = new Node("R");
Node B = new Node("B");
Node V = new Node("V");
Node A = new Node("A");
Node T = new Node("T");
Node X = new Node("X");
Node S = new Node("S");
R.LeftPointer = B;
R.RightPointer = V;
B.LeftPointer = A;
V.LeftPointer = T;
V.RightPointer = X;
T.LeftPointer = S;
preOrder(R);
static void preOrder(Node currentNode)
{
    Console.WriteLine(currentNode.Lbl);
    if (currentNode.LeftPointer != null)
    {
        preOrder(currentNode.LeftPointer);
    }
    if (currentNode.RightPointer != null)
    {
        preOrder(currentNode.RightPointer);
    }
}

class Node
{
    string label = "";
    Node? leftPtr;
    Node? rightPtr;

    public Node(string data)
    {
        label = data;
    }

    public Node LeftPointer
    {
        get => leftPtr;
        set => leftPtr = value;
    }
    public Node RightPointer
    {
        get => rightPtr;
```

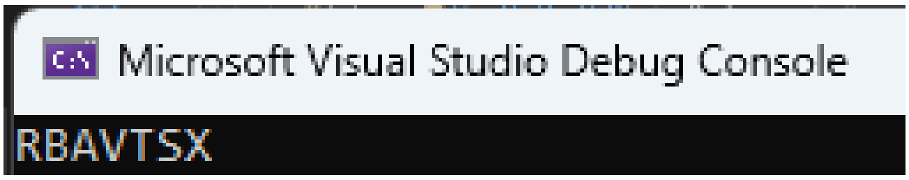
```
            set => rightPtr = value;
        }
        public String Lbl
        {
            get => label;
        }
}
```

VB.NET

```
Class Node
    Private Label As String
    Private LeftPtr As Node
    Private RightPtr As Node

    Public Sub New(data)
        Label = data
        LeftPtr = Nothing
        RightPtr = Nothing
    End Sub

    Public ReadOnly Property lbl As String
        Get
            Return Label
        End Get
    End Property
    Public Property leftPointer As Node
        Get
            Return LeftPtr
        End Get
        Set(value As Node)
            LeftPtr = value
        End Set
    End Property

    Public Property rightPointer As Node
        Get
            Return RightPtr
        End Get
        Set(value As Node)
            RightPtr = value
        End Set
    End Property
End Class
Module Program
    Sub Main(args As String())
        Dim R As New Node("R")
        Dim B As New Node("B")
        Dim V As New Node("V")
        Dim A As New Node("A")
        Dim T As New Node("T")
        Dim X As New Node("X")
        Dim S As New Node("S")
        R.leftPointer = B
        R.rightPointer = V
        B.leftPointer = A
        V.leftPointer = T
        V.rightPointer = X
        T.leftPointer = S
        PreOrder(R)
    End Sub

    Public Sub PreOrder(currentNode As Node)
        Console.WriteLine(currentNode.lbl)
        If Not currentNode.leftPointer Is Nothing
Then
            PreOrder(currentNode.leftPointer)
        End If
        If Not currentNode.rightPointer Is Nothing
Then
            PreOrder(currentNode.rightPointer)
        End If
    End Sub
End Module
```

Response A

```
Node R = new Node();

Node B = new Node();

Node A = new Node();

Node V = new Node();

Node T = new Node();

Node S = new Node();

Node X = new Node();


R.setLabel("R");

R.setLeftPtr(B);

R.setRightPtr(V);

B.setLabel("B");

B.setLeftPtr(A);

A.setLabel("A");

V.setLabel("V");

V.setLeftPtr(T);

V.setRightPtr(X);

T.setLabel("T");

T.setLeftPtr(S);

S.setLabel("S");

X.setLabel("X");

PreOrder(R);

void PreOrder(Node CurrentNode)

{

    Console.Write(CurrentNode.getLabel());

    if (CurrentNode.getLeftPtr() != null)

    {

        PreOrder(CurrentNode.getLeftPtr());

    }

    if (CurrentNode.getRightPtr() != null)

    {
        PreOrder(CurrentNode.getRightPtr());
    }
}
class Node
{
    private string Label;
    private Node leftPtr;
    private Node rightPtr;

    public void setLabel(string data)
    {
        Label = data;
    }

    public string getLabel()
    {
        return Label;
    }

    public void setLeftPtr(Node data)
    {
        leftPtr = data;
    }

    public Node getLeftPtr()
    {
        return leftPtr;
    }
    public void setRightPtr(Node data)
    {
        rightPtr = data;
    }
    public Node getRightPtr()
    {
        return rightPtr;
    }
}
```



```
RBAVTSX
```

Commentary

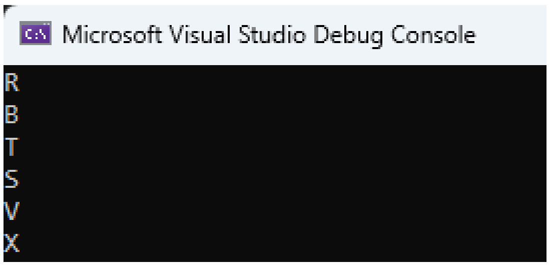
This response received **11 marks** for the code and **1 mark** for the screenshot.

This was a nearly complete response to the question. The student had created a Class for Node (MP1), however the label of each node was not assigned to the object when it was instantiated. The labels were assigned after instantiation and so MP2 was not awarded. This had not affected the student's ability to be awarded MP3 to MP12. The student had correctly set the labels and pointers to match the tree in **Figure 4** and had correctly written the PreOrder algorithm stated in **Figure 5**. This meant that **11 marks** was awarded for the program code.

The correct result was produced which was seen in the screenshot evidence, so **1 mark** was awarded for the screenshot.

Response B

```
Node R = new Node("R");

Node B = new Node("B");

Node V = new Node("V");

R.left = B;

R.right = V;

Node A = new Node("A");

Node T = new Node("T");

Node X = new Node("X");

B.left = A;

B.left = T;

V.right = X;

Node S = new Node("S");

T.left = S;

PreOrder(R);

void PreOrder(Node CurrentNode)

{

    Console.WriteLine(CurrentNode.getLabel);

    if (CurrentNode.left == null)

    {

        //DO NOTHING

    }

    else

    {

        PreOrder(CurrentNode.left);

    }

    if (CurrentNode.right == null)

    {

        //DO NOTHING

    }

    else

    {
        PreOrder(CurrentNode.right);
    }
}
class Node
{
    private string Label;
    private Node? LeftPtr;
    private Node? RightPtr;
    public Node(string label)
    {
        Label = label;
        LeftPtr = null;
        RightPtr = null;
    }

    public Node left
    {
        get => LeftPtr;
        set => LeftPtr = value;
    }

    public Node right
    {
        get => RightPtr;
        set => RightPtr = value;
    }

    public string getLabel
    {
        get => Label;
    }
}
```



```
R
B
T
S
V
X
```

Commentary

This response received **10 marks** for the code and **0 marks** for the screenshot.

This was a nearly complete response to the question. The student had correctly created the Class for Node (and included the use of ? in some attributes to account for null values). The student had also assigned labels when instantiated e.g. Node R = new Node("R");. However, the student had not correctly assigned the pointers for each node, which was evidenced by the output screenshot. It appears the student had possibly duplicated a line of code and forgot to change the node it refers to (B.left = A; and B.left = T;). This mistake meant that MP3 could not be awarded, and MP12 could also not be awarded.

The student had still correctly created the subroutine for Pre-Order based on the requirements of **Figure 5**. They had an unnecessary If-Else structure which could be simplified (with use of a ! operator in C#), but it did not affect the operation of the subroutine and so this was allowed. The evidence provided did not match the correct output from the mark scheme and so **0 marks** were given. Please note that in this example the output evidence was very helpful in awarding the correct number of marks. It is worth reminding students that even if the output is not what is expected, they should still provide the evidence even if they cannot find a solution to the problem.

Question 6

Question 6 tests Assessment Objective AO3 and requires students to create a circular queue for a set of names.

Question 6 part 6.1 and 6.2

0 6 Data about customers waiting in a line can be represented as a circular queue data structure:

- Customers leave from the front of the queue and new customers join the back of the queue (a FIFO structure)
- The first customer in line will be at the front of the queue (the Front Pointer)
- The last customer in line will be at the back of the queue (the Rear Pointer)
- There can only be a maximum of 10 customers in the queue

Task

Create a circular queue that can be used to store the name of each customer in the queue and display the queue appropriately.

The user-defined circular queue should:

- Have a one-dimensional array of 10 elements.
- Have a method `enqueue` that:
  - Takes a customer name as a parameter
  - Checks if the queue is not full, and displays an appropriate message if it is
  - If the queue is not full, it should store the customer in the queue and update the appropriate pointer
- Have a method `dequeue` that:
  - Checks if the queue is empty, and displays an appropriate message if is
  - If the queue is not empty, it should display a message saying "Customer *x* has been served" where *x* is the name of the customer leaving the queue.
  - Update any appropriate pointer
- Have a method `displayQueue` that outputs the customers in the order that they are in the queue
- Have a constructor method that sets up the queue with the customers listed in **Figure 6** in the order given.

Figure 6

FRONT Aria, Ezra, Cora, Mila, Luca, Emma, Leon, Remi **REAR**

To achieve full marks, you must make your own circular queue.

Test that your queue works by writing code that carries out these tasks in the order given:

- Enqueue Zara
- Enqueue Milo
- Enqueue Luna
- Dequeue
- Dequeue
- Enqueue Fred
- Display the Queue

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

0 6 . 1 Your PROGRAM SOURCE CODE for your entire program.

[21 marks]

0 6 . 2 The output produced when you carry out the **seven** tests.

[1 mark]

Mark scheme

Question	Part	Marking guidance	Total marks
06	1	**Maximum 20 marks** if not fully working. Students **do not** have to use an Object-Oriented approach for this question, for each language an OOP example and an alternate example (without OOP) is given for guidance. **If a built-in queue has been used only apply marks 1 to 5 and 19 to 21 to the answer** **If the user-defined queue is not circular, only apply marks 1 to 10** **For data representation:** 1. Creating an array with 10 available locations. 2. Creating integers for the front pointer and rear pointer. 3. Setting up the array to have the 8 names in the first 8 locations. 4. Setting front pointer to 1 (or 0) and setting rear pointer to 8 (or 7). 5. Using a suitable system for checking the length of the queue (this could be a variable *queueLength* or separately included in the IF statement block for enqueue and dequeue). **Note:** Depending on whether zero indexing has been used, Aria may be stored in location 0 or location 1. **For enqueue:** 6. Creating a subroutine *enqueue* with the name as the parameter. 7. Correctly checking if the queue is already full. 8. Output a suitable message if full. 9. If not full, updating the rearPointer correctly. 10. Inserting the name into the correct location (rearPointer). **For dequeue:** 11. Creating a subroutine *dequeue*. 12. Correctly checking if the queue is empty. 13. Output a suitable message if empty. 14. If not empty, correctly outputs the customer that has been served. 15. Updating the front pointer correctly. **For display:** 16. A loop is used to display the contents. 17. The loop correctly iterates from front pointer to rear pointer when the front pointer is less than the rear pointer.	**21** **AO3 = 21**

18. The loop correctly iterates from front to end of array, and from start to rear when the front pointer is greater than the rear pointer.

Note: Subroutine must be able to display queue in all circumstances of front pointer and rear pointer to gain all three marks.

Performing instructions

19. Three calls to enqueue are made with Zara, Milo and Luna each passed as the parameter.
20. Two calls to dequeue are made, followed by an enqueue with Fred as the parameter.
21. A call to display is made, and the program ends.

<u>Example Solutions:</u>

Python Example 1:

```
class Queue:
    def __init__(self):
        self.__frontPointer = 0
        self.__rearPointer = 7
        self.__queueData =
["Aria","Ezra","Cora","Mila","Luca","Emma","Leon","Remi","",""]
        self.__queueLength = 8

    def enqueue(self, name):
        if self.__queueLength == 10:
            print("Sorry the queue is full!")
        else:
            self.__rearPointer += 1
            if self.__rearPointer == 10:
                self.__rearPointer = 0
            self.__queueData[self.__rearPointer] = name
            self.__queueLength += 1

    def dequeue(self):
        if self.__queueLength == 0:
            print("Sorry the queue is empty!")
        else:
            print("Customer",self.__queueData[self.__frontPointer],
"is being served")
            self.__frontPointer += 1
            if self.__frontPointer == 10:
                self.__frontPointer = 0
            self.__queueLength -= 1

    def display(self):
        print("FRONT", end = "")
        if self.__frontPointer < self.__rearPointer:
            for i in range(self.__frontPointer,self.__rearPointer +
1):
                print("",self.__queueData[i],"",end = "")
        else:
            for i in range(self.__frontPointer, 10):
                print("",self.__queueData[i],"",end = "")
            for i in range(0, self.__rearPointer + 1):
                print("",self.__queueData[i],"",end = "")
        print("REAR")

myQueue = Queue()
myQueue.enqueue("Zara")
myQueue.enqueue("Milo")
myQueue.enqueue("Luna")
myQueue.dequeue()
myQueue.dequeue()
myQueue.enqueue("Fred")
myQueue.display()
```

Python Example 2:

```
frontPointer = 0
rearPointer = 7
queueData =
["Aria","Ezra","Cora","Mila","Luca","Emma","Leon","Remi","",""]
queueLength = 8
def enqueue(name):
    global rearPointer
    global queueLength
    if queueLength == 10:
        print("Sorry the queue is full!")
    else:
        rearPointer += 1
        if rearPointer == 10:
            rearPointer = 0
        queueData[rearPointer] = name
        queueLength += 1

def dequeue():
    global frontPointer
    global queueLength
    if queueLength == 0:
        print("Sorry the queue is empty!")
    else:
        print("Customer",queueData[frontPointer], "is being served")
        frontPointer += 1
        if frontPointer == 10:
            frontPointer = 0
        queueLength -= 1

def display():
    print("FRONT", end = "")
    if frontPointer < rearPointer:
        for i in range(frontPointer,rearPointer + 1):
            print("",queueData[i],"",end = "")
    else:
        for i in range(frontPointer, 10):
            print("",queueData[i],"",end = "")
        for i in range(0,rearPointer + 1):
            print("",queueData[i],"",end = "")
    print("REAR")

enqueue("Zara")
enqueue("Milo")
enqueue("Luna")
dequeue()
dequeue()
enqueue("Fred")
display()
```

C# Example 1:

```
Queue myQueue = new Queue();
myQueue.enqueue("Zara");
myQueue.enqueue("Milo");
myQueue.enqueue("Luna");
myQueue.dequeue();
myQueue.dequeue();
myQueue.enqueue("Fred");
myQueue.display();

class Queue
{
    private int frontPointer = 0;
    private int rearPointer = 0;
    private int queueLength = 0;
    private string[] queueData = new string[10];

    public Queue()
    {
        frontPointer = 0;
        rearPointer = 7;
        queueLength = 8;
        queueData[0] = "Aria";
        queueData[1] = "Ezra";
        queueData[2] = "Cora";
        queueData[3] = "Mila";
        queueData[4] = "Luca";
        queueData[5] = "Emma";
        queueData[6] = "Leon";
        queueData[7] = "Remi";
    }

    public void enqueue(string Name)
    {
        if (queueLength == 10)
        {
            Console.WriteLine("Sorry the queue is full");
        }
        else
        {
            rearPointer++;
            if (rearPointer == 10)
            {
                rearPointer = 1;
            }
            queueData[rearPointer] = Name;
            queueLength++;
        }
    }
    public void dequeue()
    {
        if (queueLength == 0)
        {
            Console.WriteLine("Sorry the queue is empty");
        }
        else
        {
            Console.WriteLine("Customer " + queueData[frontPointer] +
" has been served.");
            frontPointer++;
            if (frontPointer == 10)
            {
                frontPointer = 1;
            }
            queueLength -= 1;
        }
    }

    public void display()
    {
        Console.Write("FRONT");
        if (frontPointer < rearPointer)
        {
            for (int i = frontPointer; i < rearPointer; i++)
            {
                Console.Write(" " + queueData[i] + " ");
            }
        }
        else
        {
            for (int i = frontPointer; i<10;i++)
            {
                Console.Write(" " + queueData[i] + " ");
            }
            for (int i = 1; i < rearPointer; i++)
            {
                Console.Write(" " + queueData[i] + " ");
            }
        }
        Console.Write("REAR");
    }
}
```

C# Example 2:

```
int frontPointer = 0;
int rearPointer = 7;
int queueLength = 8;
string[] queueData = new string[10];
frontPointer = 0;
rearPointer = 7;
queueLength = 8;
queueData[0] = "Aria";
queueData[1] = "Ezra";
queueData[2] = "Cora";
queueData[3] = "Mila";
queueData[4] = "Luca";
```

```
queueData[5] = "Emma";
queueData[6] = "Leon";
queueData[7] = "Remi";


void enqueue(string Name)
{
    if (queueLength == 10)
    {
        Console.WriteLine("Sorry the queue is full");
    }
    else
    {
        rearPointer++;
        if (rearPointer == 10)
        {
            rearPointer = 1;
        }
        queueData[rearPointer] = Name;
        queueLength++;
    }
}

void dequeue()
{
    if (queueLength == 0)
    {
        Console.WriteLine("Sorry the queue is empty");
    }
    else
    {
        Console.WriteLine("Customer " + queueData[frontPointer] + "
has been served.");
        frontPointer++;
        if (frontPointer == 10)
        {
            frontPointer = 1;
        }
        queueLength -= 1;
    }
}

void display()
{
    Console.Write("FRONT");
    if (frontPointer < rearPointer)
    {
        for (int i = frontPointer; i < rearPointer; i++)
        {
            Console.Write(" " + queueData[i] + " ");
        }
    }
    else
    {
        for (int i = frontPointer; i < 10; i++)
        {
            Console.Write(" " + queueData[i] + " ");
        }
        for (int i = 1; i < rearPointer; i++)
        {
            Console.Write(" " + queueData[i] + " ");
        }
    }
    Console.Write("REAR");
}



enqueue("Zara");
enqueue("Milo");
enqueue("Luna");
dequeue();
dequeue();
enqueue("Fred");
display();
```

VB.NET Example 1:

```
Class Queue
    Private frontPointer As Integer
    Private rearPointer As Integer
    Private queueData(10) As String
    Private queueLength As Integer
    Public Sub New()
        frontPointer = 1
        rearPointer = 8
        queueLength = 8
        queueData(1) = "Aria"
        queueData(2) = "Ezra"
        queueData(3) = "Cora"
        queueData(4) = "Mila"
        queueData(5) = "Luca"
        queueData(6) = "Emma"
        queueData(7) = "Leon"
        queueData(8) = "Remi"
    End Sub

    Public Sub enqueue(name As String)
        If queueLength = 10 Then
            Console.WriteLine("Sorry the queue is full")
        Else
            rearPointer += 1
            If rearPointer = 11 Then
                rearPointer = 1
            End If
            queueData(rearPointer) = name
            queueLength += 1
        End If
    End Sub

    Public Sub dequeue()
        If queueLength = 0 Then
            Console.WriteLine("Sorry the queue is empty")
        Else
            Console.WriteLine("Customer " + queueData(frontPointer) &
" has been served")
            frontPointer += 1
            If frontPointer = 11 Then
```

```
                frontPointer = 1
            End If
            queueLength -= 1
        End If
    End Sub
    Public Sub display()
        Console.Write("FRONT")
        If frontPointer < rearPointer Then
            For i As Integer = frontPointer To rearPointer
                Console.Write(" " & queueData(i) & " ")
            Next
        Else
            For i As Integer = frontPointer To 10
                Console.Write(" " & queueData(i) & " ")
            Next
            For i As Integer = 1 To rearPointer
                Console.Write(" " & queueData(i) & " ")
            Next
        End If
        Console.Write("REAR")
    End Sub
End Class
Module Program
    Sub Main(args As String())
        Dim myQueue As New Queue
        myQueue.enqueue("Zara")
        myQueue.enqueue("Milo")
        myQueue.enqueue("Luna")
        myQueue.dequeue()
        myQueue.dequeue()
        myQueue.enqueue("Fred")
        myQueue.display()
    End Sub

End Module
```

VB.NET Example 2

```
Module Program
    Dim frontPointer As Integer
    Dim rearPointer As Integer
    Dim queueData(10) As String
    Dim queueLength As Integer

    Sub Main(args As String())
        frontPointer = 1
        rearPointer = 8
        queueLength = 8
        queueData(1) = "Aria"
        queueData(2) = "Ezra"
        queueData(3) = "Cora"
        queueData(4) = "Mila"
        queueData(5) = "Luca"
        queueData(6) = "Emma"
        queueData(7) = "Leon"
        queueData(8) = "Remi"
        enqueue("Zara")
        enqueue("Milo")
        enqueue("Luna")
        dequeue()
        dequeue()
        enqueue("Fred")
        display()
    End Sub


    Sub enqueue(name As String)
        If queueLength = 10 Then
            Console.WriteLine("Sorry the queue is full")
        Else
            rearPointer += 1
            If rearPointer = 11 Then
                rearPointer = 1
            End If
            queueData(rearPointer) = name
            queueLength += 1
        End If
    End Sub

    Sub dequeue()
        If queueLength = 0 Then
            Console.WriteLine("Sorry the queue is empty")
        Else
            Console.WriteLine("Customer " + queueData(frontPointer) &
" has been served")
            frontPointer += 1
            If frontPointer = 11 Then
                frontPointer = 1
            End If
            queueLength -= 1
        End If
    End Sub

   Sub display()
        Console.Write("FRONT")
        If frontPointer < rearPointer Then
            For i As Integer = frontPointer To rearPointer
                Console.Write(" " & queueData(i) & " ")
            Next
        Else
            For i As Integer = frontPointer To 10
                Console.Write(" " & queueData(i) & " ")
            Next
            For i As Integer = 1 To rearPointer
                Console.Write(" " & queueData(i) & " ")
            Next
        End If
        Console.Write("REAR")
    End Sub

End Module
```

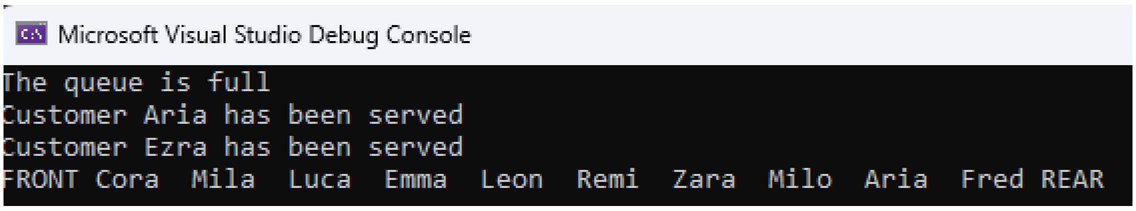
Question	Part	Marking guidance	Total marks

06	**2**	Evidence must match code from **06.1**, including prompts matching those in code. Code for 06.1 must be sensible. Test evidence shows:
		• Message to show that the queue was full when attempting to enqueue Luna.
		• Messages showing that Aria and Ezra have been served.
		• Output of queue from front to rear (words FRONT and REAR are not needed).
		
		<u>Exemplar Test Results:</u>
		`Sorry the queue is full`
		`Customer Aria has been served`
		`Customer Ezra has been served`
		`FRONT Cora  Mila  Luca  Emma  Leon  Remi  Zara Milo  Fred REAR`
			1
			
			AO3 = 1

Response A

```
Module Program

    Dim names(9) As String

    Dim FrontPointer As Integer

    Dim RearPointer As Integer

    Dim CurrentSize As Integer

    Sub Main(args As String())

        fillNames()

        enqueue("Zara")

        enqueue("Milo")

        enqueue("Luna")

        dequeue()

        dequeue()

        enqueue("Fred")

        displayQueue()

    End Sub


    Private Sub fillNames()

        names(0) = "Aria"
        names(1) = "Ezra"
        names(2) = "Cora"
        names(3) = "Mila"
        names(4) = "Luca"
        names(5) = "Emma"
        names(6) = "Leon"
        names(7) = "Remi"
        FrontPointer = 0
        RearPointer = 7
        CurrentSize = 8
    End Sub

    Sub enqueue(customerName)
        If CurrentSize = 10 Then
            Console.WriteLine("The queue is full")
        Else
            If RearPointer = 9 Then
                RearPointer = 1
            Else
                RearPointer += 1
            End If
            CurrentSize += 1
            names(RearPointer) = customerName
        End If
    End Sub

    Sub dequeue()
        If CurrentSize = 0 Then
            Console.WriteLine("The queue is empty")
        Else
            Console.WriteLine("Customer " + names(FrontPointer) + "
has been served")
            If FrontPointer = 9 Then
```

```
                FrontPointer = 1
            Else
                FrontPointer += 1
            End If
            CurrentSize -= 1
        End If
    End Sub

    Sub displayQueue()
        Console.Write("FRONT")
        If FrontPointer < RearPointer Then
            For i As Integer = FrontPointer To RearPointer
                Console.Write(" " & names(i) & " ")
            Next
        Else
            For i As Integer = FrontPointer To 9
                Console.Write(" " & names(i) & " ")
            Next
            For i As Integer = 0 To RearPointer
                Console.Write(" " & names(i) & " ")
            Next
        End If
        Console.WriteLine("REAR")
    End Sub
End Module
```



```
The queue is full
Customer Aria has been served
Customer Ezra has been served
FRONT Cora  Mila  Luca  Emma  Leon  Remi  Zara  Milo  Aria  Fred REAR
```

Commentary

This response received **21 marks** for the code and **1 mark** for the screenshot.

This was a full response to the question. The student had created an array of 10 locations (names). They created Integers for the front and rear pointer. They made use of a subroutine (fillnames()) to correctly add 8 names to the first 8 locations and also set the pointers in this subroutine. They used a variable (CurrentSize) to determine the length of the queue.

The enqueue method contained the correct parameter, it made a check on CurrentSize to determine if it was full. It output a suitable message when full. In the Else event, it updated the rear pointer correctly for a circular queue and added the name to the array in the correct location.

The dequeue method also performed all expected operations. It’s important to note that this method is the best indicator for whether the student had used a circular queue or not (based on the use of front pointer).

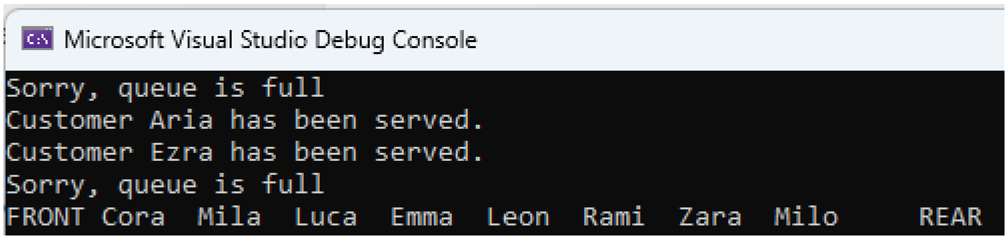
The display subroutine was implemented correctly. The If/Else events made it clear that the subroutine displayed correctly in all circumstances.

This was a good example of a non-OOP solution (OOP is not required for this question, but students may have likely learnt an OOP solution to this type of question). This resulted in **21 marks** being awarded for the program code.

The correct calls were made in the Main method, and the screenshot evidence matched the expected evidence for this question, so **1 mark** was awarded for the screenshot.

Response B

```
Class Queue

    Dim elements(10) As String

    Dim fp As Integer

    Dim rp As Integer


    Public Sub New()

        fp = 1

        rp = 8

        elements = {"", "Aria", "Ezra", "Cora", "Mila", "Luca",
"Emma", "Leon", "Rami", "", ""}

    End Sub


    Public Sub enqueue(name)

        If rp = 10 Then

            Console.WriteLine("Sorry, queue is full")

        Else

            rp += 1

            elements(rp) = name

        End If

    End Sub


    Public Sub dequeue()

        If elements(fp) = "" Then

            Console.WriteLine("Sorry, queue is empty")

        Else

            Console.WriteLine("Customer " & elements(fp) & " has
been served.")

            For i As Integer = 1 To 9

                elements(i) = elements(i + 1)

            Next

            elements(10) = ""

        End If
    End Sub


    Public Sub display()
        Console.Write("FRONT")
        For i As Integer = fp To rp
            Console.Write(" " + elements(i) + " ")
        Next
        Console.Write("REAR")
    End Sub
End Class
Module Program
    Sub Main()
        Dim q As New Queue()
        q.enqueue("Zara")
        q.enqueue("Milo")
        q.enqueue("Luna")
        q.dequeue()
        q.dequeue()
        q.enqueue("Fred")
        q.display()
    End Sub
End Module
```



```
Sorry, queue is full
Customer Aria has been served.
Customer Ezra has been served.
Sorry, queue is full
FRONT Cora  Mila  Luca  Emma  Leon  Rami  Zara  Milo     REAR
```

Commentary

This response received **12 marks** (MP1-4, MP6, MP10, MP11, MP14, MP16, MP19-21) for the code and **0 marks** for the screenshot.

This was a part-complete response to the question. The key problem with this solution was that it implemented a simple queue instead of a circular queue.

The student created an array with 10 available locations (elements). They created integers for the front pointer and rear pointer and set them to the correct values. They also set up the array to have 8 names in the first 8 locations. This student did not appear use zero-indexing and so has left a space in the first location, this was acceptable as zero-indexing was not specified in the question.

However, they did not use a suitable system for checking the length of the queue. There was evidence of the student using fp and rp to check for empty and full but this is not correct for a circular queue. This meant that MP5 could not be awarded.

The enqueue method used the correct parameter but did not correctly check if the queue was full. If rp = 10 Then would not work as the queue is circular, rp could be any value and still potentially be a full queue. For the enqueue method, only MP6 and MP10 were awarded because MP7-9 depends on the correct decision being made when checking for a full queue.

The dequeue method was marked similar to the enqueue method. Only MP11 and MP14 were awarded. This was because it did not correctly check for an empty circular queue. If elements(fp) = "" Then did not work as a queue can be empty and still contain a name in that location. Also, in the event when the queue is not empty, it did not correctly output the customer being served (the screenshot evidence proved this). The front pointer did not update correctly as it updated the contents of elements instead of simply incrementing.

The display method was awarded MP16 because it used a loop, but it did not correctly display for a circular queue and so no other marks from this section were awarded.

MP19, MP20, and MP21 were awarded as those calls were made correctly according to the question. This resulted in **12 marks** for the program code.

The screenshot evidence did not match the evidence in the mark scheme, and so was awarded **0 marks**.

Question 7

Question 7 tests Assessment Objective AO3 and requires students to create a two-player game based on ships and how they are placed.

Question 7 part 7.1 and 7.2

0 7

Ship Wars is a two-player game played on a grid of eight columns by eight rows. Player 1 places two hidden ships. These are a 5-cell dreadnought and a 30cell cruiser which are positioned either horizontally or vertically. Player 2 then attempts to locate them.

For example, in **Figure 7**, the dreadnought starts at column 2, row 2 and is horizontal. The cruiser starts at column 5, row 3, and is vertical.

Figure 7

	1	2	3	4	5	6	7	8
1	-	-	-	-	-	-	-	-
2	-	D	D	D	D	D	-	-
3	-	-	-	-	C	-	-	-
4	-	-	-	-	C	-	-	-
5	-	-	-	-	C	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-

For example, in **Figure 8**, the dreadnought starts at column 8, row 4, and is vertical. The cruiser starts at column 2, row 1, and is also vertical.

Figure 8

	1	2	3	4	5	6	7	8
1	-	C	-	-	-	-	-	-
2	-	C	-	-	-	-	-	-
3	-	C	-	-	-	-	-	-
4	-	-	-	-	-	-	-	D
5	-	-	-	-	-	-	-	D
6	-	-	-	-	-	-	-	D
7	-	-	-	-	-	-	-	D
8	-	-	-	-	-	-	-	D

Once player 1 has placed their ships, the grid will be shown to player 2 with the ships hidden.

Player 2 will then have multiple attempts to locate the ships, with a maximum of 15 guesses on the grid. Player 2 will select a cell on the grid for each move. If that cell contains part of a ship, an 'X' will appear to indicate a successful hit; otherwise, an 'O' will appear to indicate a miss.

For example, **Figure 9** shows the grid that player 2 will see. In this example, player 2 has guessed 2,3; 3,3; 5,5; and 7,8.

Figure 9

	1	2	3	4	5	6	7	8
1	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-
3	-	X	O	-	-	-	-	-
4	-	-	-	-	-	-	-	-
5	-	-	-	-	O	-	-	-
6	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	O	-

The game ends if Player 2 has destroyed both ships on the grid, or if Player 2 has used all 15 shots.

Task

Write a program that allows two players to play the ship wars game.
Your program should:

1. Use a grid with eight columns and eight rows. At the start, each location in the grid should contain "-".
2. Let player 1 place their ships, by: o Allowing them to place their dreadnought ship first, and then their cruiser ship. Stating a starting co-ordinate (column and row) for each ship and the direction (horizontal or vertical).
    - Asking them to enter this information again if:
        - The ship would reach past the edge of the grid
        - The dreadnought location overlaps with the cruiser.
3. Display the grid after both ships have been placed. They then press Enter to continue. There is no need to display the row or column headings.
4. Let player 2 guess locations on the grid. After each guess, the program must:
    - Display the grid with the ships hidden
    - Display how many guesses player 2 has left
    - Ask player 2 for a column and a row
    - If the location is already an "X" or an "O" then it should ask them to try again
    - If the location contains part of a ship, then the message "Hit!" should be displayed
    - If the location does not contain part of a ship, then the message "Miss!" should be displayed
    - If all parts of a ship are destroyed, it should say "The *x* has been destroyed!" where *x* is the name of that ship.
5. The game should end when either:
    - Player 2 has completely guessed both ships
    - Player 2 has used all 15 guesses and has not completely guessed the location of both ships.
6. A suitable message should be displayed to show whether player 1 or player 2 has won.

Test
Test that your program works by:

- Placing the dreadnought in (6,5) and choosing horizontal
- Placing the dreadnought in (6,5) and choosing vertical
- Placing the dreadnought in (4,2) and choosing horizontal
- Placing the cruiser in (4,1) and choosing vertical
- Placing the cruiser in (5,5) and choosing vertical
- Taking a SCREENSHOT of the output so far
- Play the game by guessing both ships after making at least 10 guesses
- Take a SCREENSHOT of the final grid and output

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

0 7 . 1 Your PROGRAM SOURCE CODE for your entire program.

[25 marks]

0 7 . 2 The SCREENSHOTS described in the test above.

[2 marks]

Mark Scheme

Question	Part	Marking guidance	Total marks
07	1	**Maximum 24 marks** if not fully working. **For Data Representation:** 1. Creating a two-dimensional array of 8 by 8 (**A.** zero-indexing). 2. Setting all locations on the grid a default value of "-". **For Grid Display:** 1. Loop iterates through rows. 2. Loop iterates through columns. 3. Correctly identifies when ships should be hidden and when ships should be revealed (through use of parameter or separate subroutines). **For Placing the Ships (Player One):** 4. Asking the user to enter a row and a column followed by a direction (horizontal or vertical) and inputs are assigned to appropriate variable(s). 5. Checking if the ship would go off the edge of the grid, with an appropriate output. 6. Checking if the ship would overlap with another ship, with an appropriate output. 7. Successfully looping back to the row/column entry if invalid. 8. Successfully placing the correct ship symbol into the correct locations if a correct input is given and horizontal direction is chosen. 9. Successfully placing the correct ship symbol into the correct locations if a correct input is given and vertical direction is chosen. 10. Displaying the grid with the ships revealed in the correct locations (only once when both ships have been added). **For Guessing the Ships (Player Two):** 11. Initialising a variable for remaining guesses to the value of 15. 12. Creating a condition-controlled loop until guesses remaining is zero or player two has successfully won the game. 13. Displaying the grid with the ships hidden at the start of every turn. 14. Asking for row/column from player two and the inputs are assigned to appropriate variable(s). 15. Correctly outputs "Hit!" if the location contained a ship. 16. Correctly outputs "Miss!" if the location did not contain a ship. 17. An appropriate message is displayed if the location was already a hit or a miss. 18. An appropriate message is displayed when a ship has been completely guessed.	25 AO3 = 25

19. Program correctly outputs the winner of the game.

For Program Structure:

20. At least one user-defined subroutine created and called, which has an appropriate meaningful name.
21. Appropriate overall division of program into subroutines. **Note:** Must be at least three programmer-created subroutines.
22. No unnecessary repetition of code to achieve the same purpose in more than one place. For example, the code to display the board is not duplicated for every turn. **Note:** Some attempt must have been made to write code for both players to award this mark.
23. No use of global variables, all values passed between subroutines using parameters and return values.
24. Sensible use of identifiers throughout.

Example Solutions:

Python

```
def placeShips():
    grid = [["-" for col in range(8)]for row in range(8)]
    dreadnoughtPlaced = False
    cruiserPlaced = False
    print("Player 1 you may now start placing the ships on
the grid")
    while dreadnoughtPlaced == False:
        grid,dreadnoughtPlaced =
enterShipIntoGrid(grid,"Dreadnought",5,"D")
    while cruiserPlaced == False:
        grid,cruiserPlaced =
enterShipIntoGrid(grid,"Cruiser",3,"C")
    displayGrid(grid,False)
    return grid

def displayGrid(grid,hidden):
    for i in range(8):
        for j in range(8):
            if hidden:
                if grid[i][j] == "C" or grid[i][j] == "D":
                    print("-",end = "")
                else:
                    print(grid[i][j], end = "")
            else:
                print(grid[i][j], end = "")
        print()


def enterShipIntoGrid(grid,shipName,shipLength,shipSym):
    shipX = -1
    shipY = -1
    print("Player 1 please choose where the",shipName,"shall
be placed")
    shipX = int(input("Enter column")) - 1 #They enter 1 to 8
    shipY = int(input("Enter row")) - 1
    shipDirection = input("enter h or v (horizontal or
vertical)")
    if shipDirection == "h":
        for i in range(shipLength):
            if shipX + i > 7:
                print("That location would go past the edge,
please try again")
                return grid,False
            elif grid[shipY][shipX + i] != "-":
                print("That location overlaps with another
ship, please try again")
                return grid,False
        for i in range(shipLength):
            grid[shipY][shipX+i] = shipSym
    else:
        for i in range(shipLength):
            if shipY + i > 7:
                print("That location would go past the edge,
please try again")
                return grid,False
            elif grid[shipY+i][shipX] != "-":
                print("That location overlaps with another
ship, please try again")
                return grid,False
        for i in range(shipLength):
            grid[shipY+i][shipX] = shipSym
    return grid,True

def attackShips(grid):
    shots = 15
    dreadnoughtCells = 5
    cruiserCells = 3
    gameOver = False
    shotX = -1
    shotY = -1
    print("Player 2 you may now start firing shots at the
grid")
    while shots > 0 and gameOver == False:
        displayGrid(grid,False)
        print("You have",shots,"shots remaining")
        shotX = int(input("enter column")) - 1
        shotY = int(input("enter row")) - 1
        if grid[shotY][shotX] == "O" or grid[shotY][shotX] ==
"X":
            print("You have already shot there, please try
again")
        else:
            if grid[shotY][shotX] == "D":
                grid[shotY][shotX] = "X"
                print("That's a hit!")
                dreadnoughtCells -=1
                if dreadnoughtCells == 0:
                    print("The Dreadnought has been
destroyed!")
            elif grid[shotY][shotX] == "C":
                grid[shotY][shotX] = "X"
                print("That's a hit!")
                cruiserCells -=1
                if cruiserCells == 0:
                    print("The Cruiser has been destroyed!")
            else:
                grid[shotY][shotX] = "O"
                print("That's a miss!")
            shots -= 1
            if dreadnoughtCells == 0 and cruiserCells == 0:
                gameOver = True
    if gameOver == True:
        print("Player 2 you have won!")
    else:
        print("Player 1 you have won!")

grid = placeShips()
attackShips(grid)
```

C#

```
string[,] grid = placeShips();
attackShips(grid);

void attackShips(string[,] grid)
{
    int shots = 15;
    int dreadnoughtCells = 5;
    int cruiserCells = 3;
    bool gameOver = false;
    int shotX = -1;
    int shotY = -1;
    Console.WriteLine("Player 2 you may now start firing
shots at the grid");
    do
    {
        displayGrid(grid, true);
        Console.WriteLine("You have " + shots + " shots
remaining");
        Console.WriteLine("enter column");
        shotX = Convert.ToInt16(Console.ReadLine()) - 1;
        Console.WriteLine("enter row");
        shotY = Convert.ToInt16(Console.ReadLine()) - 1;
        if (grid[shotX,shotY] == "O" || grid[shotX,shotY] ==
"X")
        {
            Console.WriteLine("You have already shot there,
please try again!");
        }
        else
        {
            if (grid[shotX,shotY] == "D")
            {
                grid[shotX, shotY] = "X";
                itsAHit(ref dreadnoughtCells, "Dreadnought");
            }
            else if (grid[shotX,shotY] == "C")
            {
                grid[shotX, shotY] = "X";
                itsAHit(ref cruiserCells, "Cruiser");
            }
            else
            {
                grid[shotX, shotY] = "O";
                Console.WriteLine("That's a miss!");
            }
            shots--;
            if (dreadnoughtCells == 0 && cruiserCells == 0)
            {
                gameOver = true;
            }
        }

    } while (shots > 0 && gameOver == false );
    if (gameOver == true)
    {
        Console.WriteLine("Player 2 you have won!");
    }
    else
    {
        Console.WriteLine("Player 1 you have won!");
    }
}

void itsAHit(ref int shipCells, string shipName)
{
    Console.WriteLine("That's a hit!");
    shipCells--;
    if (shipCells == 0)
    {
        Console.WriteLine("The " + shipName + " has been
destroyed!");
    }
}
string[,] placeShips()
{
    string[,] grid = new string[8,8];
    bool dreadnoughtPlaced = false;
    bool cruiserPlaced = false;
    setUpGrid(ref grid);
    do
    {
        dreadnoughtPlaced = enterShipIntoGrid(ref grid,
"Dreadnought", 5, "D");
    } while (dreadnoughtPlaced == false);
    do
    {
        cruiserPlaced = enterShipIntoGrid(ref grid,
"Cruiser", 3, "C");
    } while (cruiserPlaced == false);
    displayGrid(grid, false);
    Console.WriteLine("Your ships are placed. Press Enter to
continue");
    Console.ReadLine();
    return grid;
}

void setUpGrid(ref string[,] grid)
{
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            grid[i, j] = "-";
        }
    }
}
void displayGrid(string[,] grid, bool hidden)
{
    for (int j = 0;j<8;j++)
    {
        for (int i = 0;i<8;i++)
        {
            if (hidden)
            {
                if (grid[i,j] == "C" || grid[i, j] == "D")
                {
                    Console.Write("-");
                }
                else
                {
                    Console.Write(grid[i, j]);
                }
            }
            else
            {
                Console.Write(grid[i, j]);
            }
        }
        Console.WriteLine();
    }
    Console.WriteLine();
}
bool enterShipIntoGrid(ref string[,] grid, string shipName,
int shipLen, string shipSym)
{
    int shipX = -1;
    int shipY = -1;
    Console.WriteLine("Player 1 please choose where the " +
shipName + " shall be placed");
    Console.WriteLine("enter column");
    shipX = Convert.ToInt16(Console.ReadLine()) - 1; //user
types in 1 to 8
    Console.WriteLine("enter row");
    shipY = Convert.ToInt16(Console.ReadLine()) - 1;
    string shipDirection = "";
    Console.WriteLine("enter h or v");
    shipDirection = Console.ReadLine();
    if (shipDirection == "h")
    {
        for(int i = 0;i < shipLen;i++)
        {
            if (shipX + i > 7)
            {
                Console.WriteLine("That location would go
past the edge, please try again");
                return false;
            }
            else if (grid[shipX+i,shipY] != "-")
            {
                Console.WriteLine("That location overlaps
with another ship, please try again");
                return false;
            }
        }
        for(int i = 0; i < shipLen; i++)
        {
            grid[shipX + i, shipY] = shipSym;
        }
    }
    else
    {
        for (int i = 0; i < shipLen; i++)
        {
            if (shipY + i > 7)
            {
                Console.WriteLine("That location would go
past the edge, please try again");
                return false;
            }
            else if (grid[shipX, shipY+i] != "-")
            {
                Console.WriteLine("That location overlaps
with another ship, please try again");
                return false;
            }
        }
        for (int i = 0; i < shipLen; i++)
        {
            grid[shipX, shipY+i] = shipSym;
        }
    }
    return true;
}
```

VB.NET

```
Module Program
    Sub Main(args As String())
        Dim grid(8, 8) As String
        setUpGrid(grid)
        placeShips(grid)
        attackShips(grid)
    End Sub

    Private Sub attackShips(grid)
        Dim shotsRemaining As Integer = 15
        Dim dreadnoughtCells As Integer = 5
        Dim cruiserCells As Integer = 3
        Dim gameOver As Boolean = False
        Dim shotX As Integer
        Dim shotY As Integer
        Console.WriteLine("Player 2 you may now start firing
shots at the grid")
        Do Until shotsRemaining = 0 Or gameOver = True
```

```
            displayGrid(grid, True)
            Console.WriteLine("You have " & shotsRemaining &
" shots remaining...")
            Console.WriteLine("enter column")
            shotX = Console.ReadLine
            Console.WriteLine("enter row")
            shotY = Console.ReadLine
            If grid(shotX, shotY) = "O" Or grid(shotX, shotY)
= "X" Then
                Console.WriteLine("You have already shot
there, please try again")
            Else
                If grid(shotX, shotY) = "D" Then
                    grid(shotX, shotY) = "X"
                    Console.WriteLine("That's a hit!")
                    dreadnoughtCells -= 1
                    If dreadnoughtCells = 0 Then
                        Console.WriteLine("The Dreadnought
has been destroyed!")
                    End If
                ElseIf grid(shotX, shotY) = "C" Then
                    grid(shotX, shotY) = "X"
                    Console.WriteLine("That's a hit!")
                    cruiserCells -= 1
                    If cruiserCells = 0 Then
                        Console.WriteLine("The Cruiser has
been destroyed!")
                    End If
                Else
                    grid(shotX, shotY) = "O"
                    Console.WriteLine("That's a miss!")
                End If
                shotsRemaining -= 1
                If dreadnoughtCells = 0 And cruiserCells = 0
Then
                    gameOver = True
                End If
            End If
        Loop
        If gameOver = True Then
            Console.WriteLine("Player 2 you have won!")
        Else
            Console.WriteLine("Player 1 you have won!")
        End If
    End Sub

    Private Sub displayGrid(grid, hidden)
        For j As Integer = 1 To 8
            For i As Integer = 1 To 8
                If hidden Then
                    If grid(i, j) = "C" Or grid(i, j) = "D"
Then
                        Console.Write("-")
                    Else
                        Console.Write(grid(i, j))
                    End If
                Else
                    Console.Write(grid(i, j))
                End If

            Next
            Console.WriteLine()
        Next
        Console.WriteLine()
    End Sub

    Private Sub placeShips(grid)
        Dim dreadnoughtPlaced As Boolean = False
        Dim cruiserPlaced As Boolean = False
        Console.WriteLine("Player 1 you may now start placing
the ships on the grid")
        Do Until dreadnoughtPlaced
            dreadnoughtPlaced = enterShipIntoGrid(grid,
"Dreadnought", 5, "D")
        Loop
        Do Until cruiserPlaced
            cruiserPlaced = enterShipIntoGrid(grid,
"Cruiser", 3, "C")
        Loop
        displayGrid(grid, False)
        Console.WriteLine("Your ships are placed. Press Enter
to continue")
        Console.ReadLine()
        Console.Clear() 'THIS LINE IS NOT NEEDED
    End Sub

    Function enterShipIntoGrid(grid, shipName, lengthOfShip,
shipSymbol) As Boolean
        Dim shipX As Integer
        Dim shipY As Integer
        Dim shipDirection As String
        Console.WriteLine("Player 1 please choose where the "
+ shipName + " shall be placed")
        Console.WriteLine("enter column")
        shipX = Console.ReadLine
        Console.WriteLine("enter row")
        shipY = Console.ReadLine
        Console.WriteLine("enter h or v (horizontal or
vertical)")
        shipDirection = Console.ReadLine.ToUpper()
        If shipDirection = "H" Then
            'Check X Co-ordinate
            For i As Integer = 0 To lengthOfShip - 1
                If shipX + i > 8 Then
                    Console.WriteLine("That location would go
past the edge, please try again")
                    Return False
                ElseIf grid(shipX + i, shipY) <> "-" Then
                    Console.WriteLine("That location overlaps
with another ship, please try again")
                    Return False
                End If
            Next
            For i As Integer = 0 To lengthOfShip - 1
                grid(shipX + i, shipY) = shipSymbol
            Next
```

oxfordaqa.com

```
        Else
            'Check Y Co-Ordinate
            For i As Integer = 0 To lengthOfShip - 1
                If shipY + i > 8 Then
                    Console.WriteLine("That location would go past the edge, please try again")
                    Return False
                ElseIf grid(shipX, shipY + i) <> "-" Then
                    Console.WriteLine("That location overlaps with another ship, please try again")
                    Return False
                End If
            Next
            For i As Integer = 0 To lengthOfShip - 1
                grid(shipX, shipY + i) = shipSymbol
            Next
        End If

        Return True
    End Function
    Private Sub setUpGrid(grid)
        For i As Integer = 1 To 8
            For j As Integer = 1 To 8
                grid(i, j) = "-"
            Next
        Next
    End Sub



End Module
```

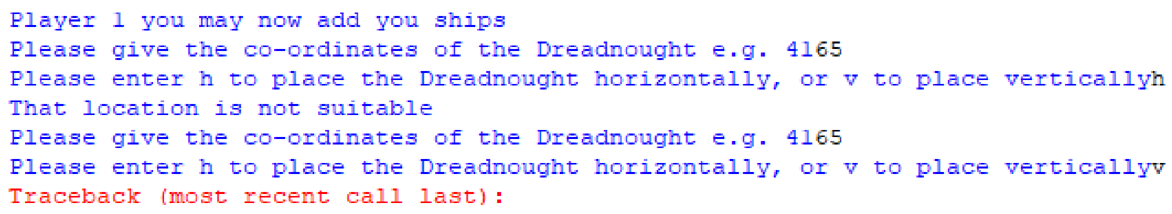
Question	Part	Marking guidance	Total marks

07	2	Evidence must match code from **07.1**, including prompts matching those in code. Code for 07.1 must be sensible.
		
		Screenshot one (**1 mark**) must show:
		• Appropriate message for (6,5) horizontal placement
		• Appropriate message for (6,5) vertical placement
		• Valid placement for (4,2) horizontal
		• Appropriate message for (4,1) vertical placement
		• Valid placement for (5,5) vertical
		• Grid displayed with two ships in correct positions
		
		Screenshot two (**1 mark**) must show:
		• Final grid (**A.** grid before final shot is made).
		Note: the "O" locations will vary depending on their guesses but the "X" locations will always be the same.
		• A message showing how many guesses are remaining (must be 5 or lower).
		• A message showing that a ship has been completely guessed.
		• A message stating that player two has won.
		
		<u>**Exemplar Test Results:**</u>
		
		`SCREENSHOT ONE`
		`Player 1 you may now start placing the ships on the grid`
		`Player 1 please choose where the Dreadnought shall be placed`
		`enter column: 6`
		`enter row: 5`
		`enter h or v: h`
		`That location would go past the edge, please try again`
		`Player 1 please choose where the Dreadnought shall be placed`
		`enter column: 6`
		`enter row: 5`
		`enter h or v: v`
		`That location would go past the edge, please try again`
		`Player 1 please choose where the Dreadnought shall be placed`
		`enter column 4`
		`enter row 2`
		`enter h or v: h`
		`Player 1 please choose where the Cruiser shall be placed`
		`enter column: 4`
		`enter row: 1`
		`enter h or v: v`
		`That location overlaps with another ship, please try again`
		`Player 1 please choose where the Cruiser shall be placed`
		`enter column: 5`
			2
			
			AO3 = 2

```
enter row: 5
enter h or v: v
--------
---DDDDD
--------
--------
----C---
----C---
----C---
--------

Your ships are placed. Press Enter to continue
SCREENSHOT TWO
O--O----
--OXXXX-
--------
---O----
----X---
---OX---
----X---
--------

You have 3 shots remaining...
enter column
8
enter row
2
That's a hit!
The Dreadnought has been destroyed!
Player 2 you have won!
```

Response A

```
def addShips(grid):

    print("Player 1 you may now add you ships")

    valid = False

    while valid == False:

        dLoc = input("Please give the co-ordinates of the
Dreadnought e.g. 41")

        dDirection = input("Please enter h to place the Dreadnought
horizontally, or v to place vertically")

        valid = True

        for i in range(5):

            if dDirection == "h":

                if int(dLoc[0])-1 + i > 8:

                    print("That location is not suitable")

                    valid = False

            elif dDirection == "v":

                if int(dLoc[1])-1 + i > 8:
                    print("That location is not suitable")
                    valid = False
    for i in range(5):
        if dDirection == "h":
            grid[int(dLoc[0])-1+i][int(dLoc[1])-1] = "D"
        elif dDirection == "v":
            grid[int(dLoc[0])-1][int(dLoc[1])-1+i] = "D"
    valid = False
    while valid == False:
        cLoc = input("Please give the co-ordinates of the Cruiser
e.g. 41")
        cDirection = input("Please enter h to place the Cruiser
horizontally, or v to place vertically")
        valid = True
        for i in range(3):
            if cDirection == "h":
                if int(cLoc[0]) -1 + i > 8:
                    print("That location is not suitable")
                    valid = False
                elif grid[int(cLoc[0]) -1 + i][int(cLoc[1]) -1] ==
"D":
                    print("This ship overlaps with the Dreadnought")
                    valid = False
            elif cDirection == "v":
                if int(cLoc[1]-1) + i > 8:
                    print("That location is not suitable")
                    valid = False
                elif grid[int(cLoc[0])-1][int(cLoc[1]) -1 + i] ==
"D":
                    print("This ship overlaps with the Dreadnought")
                    valid = False
    for i in range(3):
        if cDirection == "h":
            grid[int(cLoc[0])-1+i][int(cLoc[1])-1] = "C"
        elif cDirection == "v":
            grid[int(cLoc[0])-1][int(cLoc[1])-1+i] = "C"


def displayGrid(grid, hidden):
    print()
    for j in range(8):
        for i in range(8):
            if hidden:
                if grid[i][j] == "D" or grid[i][j] == "C":
                    print("-", end = "")
            else:
                print(grid[i][j], end = "")
        print()
    print()


grid = [["-","-","-","-","-","-","-","-"],["-","-","-","-","-","-
","-","-"],["-","-","-","-","-","-","-","-"],["-","-","-","-","-","-
","-","-"],["-","-","-","-","-","-","-","-"],["-","-","-","-","-","-
","-","-"],["-","-","-","-","-","-","-","-"],["-","-","-","-","-","-
","-","-"]]
grid = addShips(grid)
displayGrid(grid, False)
```



```
Player 1 you may now add you ships
Please give the co-ordinates of the Dreadnought e.g. 4165
Please enter h to place the Dreadnought horizontally, or v to place verticallyh
That location is not suitable
Please give the co-ordinates of the Dreadnought e.g. 4165
Please enter h to place the Dreadnought horizontally, or v to place verticallyv
Traceback (most recent call last):
```

Commentary

This response received **11 marks** (MP1-9, MP20, MP23) for the code and **0 marks** for the screenshot.

This was a partially complete response. The code (as evidenced by the screenshot) appeared to be incomplete, possibly because they ran out of time. There was screenshot evidence provided. To justify where the marks were given, the commentary has been split into sections.

Data representation

(MP1) The student created a two-dimensional array (grid) with the correct number of locations.

(MP2) All values were initialised to "-".

Grid display

(MP3) A loop (j) iterated through 8 values .

(MP4) A loop (i) is nested within the first loop which iterated through 8 values.

(MP5) They used a parameter (hidden) to represent whether it should display ships as hidden or as normal.

For placing the ships (Player One)

(MP6) Student asked for row and column together (e.g. dLoc) and direction (e.g. dDirection).

(MP7) Evidence of code checking if the ship goes off the edge of the grid

```
for i in range(3):
    if cDirection == "h":
        if int(cLoc[0]) -1 + i > 8:
            print("That location is not suitable")
            valid = False
```

(MP8) Evidence of code checking if the ship overlaps with another ship

```
    elif grid[int(cLoc[0]) -1 + i][int(cLoc[1]) -1] == "D":
        print("This ship overlaps with the Dreadnought")
        valid = False
```

(MP9) Student contained code in iteration structures which looped again until a Boolean data structure (valid) was True. This Boolean correctly stayed True when passing all validation checks. There was also evidence of the code looping back in the screenshot evidence.

MP10-MP12 were not awarded. There was code which supported these mark points, however they did not work successfully and there was no supporting evidence to show this in the screenshot. The screenshot appeared to show an error occurring which provided further evidence that this was not completed successfully.

For attacking the ships (Player Two)

There was no code provided for this part of the program and so none of the mark points from this section were awarded.

For program structure:

(MP20) There were examples of two user-defined subroutines with meaningful names.

MP21 was not awarded as there needed to be at least three user-defined subroutines.

MP22 was not as there was no code for Player Two in this response.

(MP23) All values were passed appropriately between subroutines. grid was declared as a top-level statement but whenever it was used in a subroutine it was passed as a parameter, so this was accepted.

This meant that the program code was awarded **11 marks**. The screenshot evidence did not match the evidence provided in the mark scheme and so this received **0 marks**.

Response B

```
def playGame(grid):
    shots = 15
    shipsDestroyed = False
    while shipsDestroyed == False and shots > 0:
        showGrid(grid)
        location = input("Please type a location to fire at")
        x = int(location[0]) - 1
        y = int(location[1]) - 1
        if grid[x][y] == "D" or grid[x][y] == "C":
            print("It's a hit!")
            grid[x][y] = "X"
        elif grid[x][y] == "X" or grid[x][y] == "O":
            print("You have already chosen this location")
        else:
            print("It's a miss!")
            grid[x][y] = "O"
        shots -= 1
        shipsDestroyed = True
        for i in range(8):
            for j in range(8):
                if grid[i][j] == "D" or grid[i][j] == "C":
                    shipsDestroyed = False
    print("Game over!")
    if shipsDestroyed == True:
        print("Player 2 wins!")
    else:
        print("Player 1 wins!")


def showGrid(grid):
    print()
    for i in range(8):
        for j in range(8):
            print(grid[i][j], end = "")
        print()
    print()


def placeShips(grid):
    dreadnought = input("Please type the xy coordinate and direction
for example 42h")
    dreadX = int(dreadnought[0]) - 1
    dreadY = int(dreadnought[1]) - 1
    dreadHV = dreadnought[2]
    try:
        if dreadHV == "h":
            for i in range(5):
                grid[dreadX + i][dreadY] = "D"
        else:
            for i in range(5):
                grid[dreadX][dreadY + i] = "D"
    except:
        print("Sorry that is not valid")
    cruiser = input("Please type the xy coordinate and direction for
example 42h")
```

```
    cruiseX = int(cruiser[0]) - 1

    cruiseY = int(cruiser[1]) - 1

    cruiseHV = cruiser[2]

    try:

        if cruiseHV == "h":

            for i in range(5):

                grid[cruiseX + i][cruiseY] = "C"

        else:

            for i in range(5):

                grid[cruiseX][cruiseY + i] = "C"

    except:

        print("Sorry that is not valid")

    return grid


def main():

    grid = [["-","-","-","-","-","-","-","-"],["-","-","-","-","-
","-","-","-"],["-","-","-","-","-","-","-","-"],["-","-","-","-","-
","-","-","-"],["-","-","-","-","-","-","-","-"],["-","-","-","-","-
","-","-","-"],["-","-","-","-","-","-","-","-"],["-","-","-","-","-
","-","-","-"]]

    grid = placeShips(grid)

    showGrid(grid)

    playGame(grid)


if __name__ == "__main__":

    main()
```

SCREENSHOT ONE

```
Please type the xy coordinate and direction for example 42h65h
Sorry that is not valid
Please type the xy coordinate and direction for example 42h65v
Sorry that is not valid

--------
--------
--------
--------
--------
----CCCC
----D---
----D---
```

SCREENSHOT TWO

```
Please type a location to fire at65
It's a hit!

O-------
-O------
--O-----
---O----
----O---
----XXXX
----D---
----DO--

Please type a location to fire at75
It's a hit!

O-------
-O------
--O-----
---O----
----O---
----XXXX
----X---
----DO--

Please type a location to fire at85
It's a hit!
Game over!
Player 2 wins!
```

Commentary

This response received **18 marks** (MP1-MP4, MP6, MP7,MP12-14,MP16-19,MP21-25) for the code and **0 marks** for the screenshot.

This was also a part response to the question. The student had completed a mix of objectives for both players but missed out some key parts to the program. The commentary has been split into sections:

For data representation

(MP1) The student created a 2-dimensional array (grid) with the correct number of locations.

(MP2) The student initialised all locations of the array with the correct value ("-").

For grid display

(MP3) A loop (i) iterated through 8 values.

(MP4) A loop (j) was nested within the first loop and iterated through 8 values.

MP5 was not awarded because there was no option to hide the ships from the grid (it will always print the contents of grid).

For placing the ships (Player One)

(MP6) The student had combined the input for the location and the direction and stored this in a suitable variable (dreadnought). The student had then split up the data into 3 separate variables.

(MP7) The student used exception handling to check if the ship went off the edge of the array.

MP8-11 were not awarded because there was no loop checking for incorrect placements. There was also no selection structure checking for the overlapping of ships. It placed ships in locations which were not valid (the evidence in the screenshot also proved this).

(MP12) The grid was displayed with the ships in the chosen locations once both ships had been added (the evidence in the screenshot also proved this).

For attacking the ships (Player Two)

(MP13) A variable (shots) was initialised to 15.

(MP14) A loop is created which ended when all ships had been destroyed or shots = 0.

MP15 was not awarded as there was no option to display the grid with the ships hidden.

(MP16) A location was inputted and stored in a variable (location). This was then split up into two variables (x and y).

(MP17) Program outputted a correct message when a ship was hit.

(MP18) Program outputted a correct message when a ship was missed.

(MP19) Program outputted a correct message when a hit or miss had already taken place in that location.

MP20 was not awarded as there was no check for when a ship had been destroyed (only when all ships have been destroyed in a nested-iteration structure).

(MP21) The logic of the program showed that the correct winner would be output (the screenshot evidence also showed the correct result when all ships had been destroyed).

For program structure

(MP22) There were 4 user-defined subroutines with suitable names.

(MP23) The division of the program was appropriate (playGame, showGrid, and placeShips are all sensible procedures to use in this example). main was not required, top-level statements were accepted.

(MP24) There was no unnecessary duplication, showGrid for example was simply called when required and its code was not copied in multiple places.

(MP25) There were no global variables, variables were passed between subroutine calls (e.g. grid as a parameter).

This meant that **18 marks** in total were awarded for the program code. The screenshot evidence was very helpful to award marks for the code (and this is worth noting), however it did not meet the requirements of the mark scheme evidence (e.g. screenshot two did not show that a single ship had been destroyed) and so was awarded **0 marks.**