6(a)

1 mark for correct answer

Type of compression: lossy
Justification: There will be fewer samples per second, so data will be permanently lost

// There will be fewer samples per second, so the original sound cannot be re-created

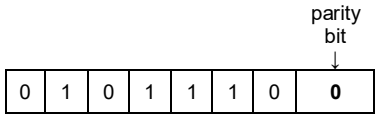
6(b)

1 mark for correct words in the correct order

Binary Value	0011 1000	0011 1100	0011 1110
Word	Science	is	Amazing!

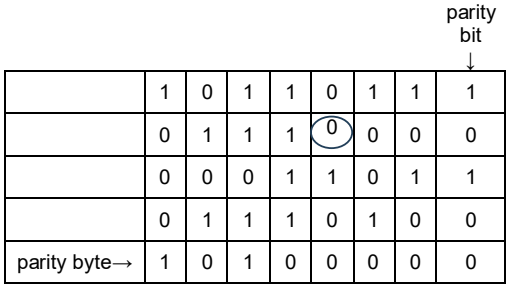
6(c)(i)

1 mark for correct answer



6(c)(ii)

1 mark for correct answer



[1]

6(d)

1 mark for the working
1 mark for the correct answer

Working:
(1000 * 2000 * 16) / (8 * 1000 * 1000)
// (1000 * 2000 * 2 / (1000 * 1000)

Answer: 4 megabytes

2(a)

1 mark each

- `Queue` declared as a (global) 1D array of 100 string elements all initialised to `""`
- `QueueHead` and `QueueTail` initialised with `-1`, `NumberItems` initialised to `0`

Example program code

Java

```
public static String[] Queue = new String[100];
public static Integer QueueHead;
public static Integer QueueTail;
public static Integer NumberItems;
for(int x = 0; x < 100; x++){
     Queue[x] = "";
}
QueueHead = -1;
QueueTail = -1;
NumberItems = 0;
```

VB.NET

```
    Dim Queue(99) As String
    Dim QueueHead As Integer
    Dim QueueTail As Integer
    Dim NumberItems As Integer
For x = 0 To 99
     Queue(x) = ""
 Next
 QueueHead = -1
 QueueTail = -1
 NumberItems = 0
```

2(a)

Python

```
global Queue, QueueHead, QueueTail, NumberItems
Queue = []
    for x in range(100):
        Queue.append("")
    QueueHead = -1
    QueueTail = -1
    NumberItems = 0
```

2(b)

1 mark each

- Function header (and end where appropriate) taking one (string) parameter **and** checking full **and** returning `FALSE`
- (otherwise) Storing parameter in `QueueTail + 1`
- Incrementing `QueueTail` and `NumberItems`
- (Dealing with first element) If `QueueHead = -1` store `0` in `QueueHead` / increment `QueueHead`
- Return `TRUE` in **all** cases when inserted

Example program code.

Java

```
public static Boolean Enqueue(String TheData){
      if(QueueHead == -1){
            Queue[0] = TheData;
            QueueHead = 0;
            QueueTail = 0;
            NumberItems++;
            return true;
      }else if(QueueTail >= 99){
            return false;
      }else{
            Queue[QueueTail+1] = TheData;
            QueueTail++;
            NumberItems++;
            return true;
      }

}
```


2(b)

VB.NET

```
Function Enqueue(TheData)
    If QueueHead = -1 Then
        Queue(0) = TheData
        QueueHead = 0
        QueueTail = 0
        NumberItems += 1
        Return True
    ElseIf QueueTail >= 99 Then
        Return False
    Else
        Queue(QueueTail + 1) = TheData
        QueueTail += 1
        NumberItems += 1
        Return True
    End If
End Function
```

Python

```
def Enqueue(TheData):
    global Queue, QueueHead, QueueTail, NumberItems
    if(QueueHead == -1){
         Queue[0] = TheData
         QueueHead = 0
         QueueTail = 0
         NumberItems +=1
         return True
    elif QueueTail >= 99:
        return False
    else:
        Queue[QueueTail+1] = TheData
        QueueTail +=1
        NumberItems +=1
        return True
```

2(c)

1 mark each

- Function header (and end), checking if `Queue` is empty (`NumberItems = 0` or `QueueHead > QueueTail`) **and** returning "`False`"
- (otherwise) returning `Queue[QueueHead]`
- Incrementing `QueueHead`, decrementing `NumberItems`

Example program code

Java

```
public static String Dequeue(){
      String ReturnValue;
      if(NumberItems == 0){
            return "False";
      }else{
            ReturnValue = Queue[QueueHead];
            QueueHead++;
            NumberItems--;
            return ReturnValue;
      }
}
```

VB.NET

```
Function Dequeue()

    If NumberItems = 0 Then
        Return "False"
    Else
        Dequeue = Queue(QueueHead)
        QueueHead += 1
        NumberItems -= 1

    End If
End Function
```


2(c)

Python

```
def Dequeue():
    global Queue, QueueHead, QueueTail, NumberItems
    if NumberItems == 0:
        return "False"
    else:
        ReturnData = Queue[QueueHead]
        QueueHead += 1
        NumberItems -=1
        return ReturnData
```

2(d)

1 mark each to max 5

- Procedure header (and end where appropriate)
- Opening the file **and** closing the file (in appropriate place)
- Looping until EOF // looping through each line …
- … read in each value …
- … calling `Enqueue()` with read in value and store/use return value
- Exception handling with try except and appropriate output with all file access within try

Example program code

Java

```
public static void ReadData(){
            Boolean ReturnValue;
            Boolean FinishLoop = false;
            try{
                  Scanner Scanner1 = new Scanner(new File("BinaryData.txt"));
                  while(Scanner1.hasNextLine() && FinishLoop == false){
                        ReturnValue = Enqueue(Scanner1.nextLine());
                        if(ReturnValue.equals("False")){
                              FinishLoop = true;
                        }
                  }
                  Scanner1.close();
            } catch(FileNotFoundException ex){
                  System.out.println("No file found");
            }
      }
```


2(d)

VB.NET

```
Sub ReadData()
     Dim ReturnValue As String
     Dim DataReader As New System.IO.StreamReader("BinaryData.txt")
     Dim FinishLoop As Boolean = True
     Do Until DataReader.EndOfStream Or FinishLoop = False
         ReturnValue = Enqueue(DataReader.ReadLine())
         If ReturnValue = False Then
             FinishLoop = False
         End If
     Loop
     DataReader.Close()
 End Sub
```

Python

```
def ReadData():

    TheFile = open("BinaryData.txt")
    for Line in TheFile:
        ReturnValue = Enqueue(Line.strip())
        if ReturnValue == False:
            break

    TheFile.close()
```

2(e)

1 mark each

- Procedure header (and end where appropriate), storing final string compressed data in global variable
- Call `Dequeue()` **and** storing return value …
- … repeatedly until the return value == "False"
- Comparing new value with previous …
- … maintaining counter of number of occurrences …
- … appending digit and number of occurrences to a string without overriding

Example program code

Java

```
public static void Compress(){
            String First = Dequeue();
            String NewLine = "";
            Integer Count;
            String NextChar;
            while(NumberItems > 0 && First != "False"){
                  Count = 1;
                  NextChar = Dequeue();
                  while(NextChar.equals(First)){
                        Count++;
                        First = NextChar;
                        NextChar = Dequeue();
                  }
                  NewLine = First + Count;
                  NewString = NewString + NewLine;
                  First = NextChar;
            }}
```


2(e)

VB.NET

```
Sub Compress()
     Dim First As String = Dequeue()
     Dim NewLine As String = ""
     Dim Count As Integer
     Dim NextChar As String
     While NumberItems > 0 And First <> "False"
         Count = 1
         NextChar = Dequeue()

         While NextChar = First
             Count += 1
             First = NextChar
             NextChar = Dequeue()
         End While

         NewLine = First & Count
         NewString = NewString & NewLine
         First = NextChar
     End While
```

Python

```
def Compress():
    global NewString
    First = Dequeue()
    NewString = ""
    while NumberItems > 0 and First != "False":
        Count = 1
        NextChar = Dequeue()
        while NextChar == First:
            Count += 1
            First = NextChar
            NextChar = Dequeue()
        NewLine = First + str(Count)
        NewString = NewString + NewLine
        First = NextChar
```

2(f)(i)

1 mark each
- Calling `ReadData()` then `Compress()`
- Outputting compressed string

Example program code

Java
```
public static void main(String args[]){
           NewString = "";
           for(int x = 0; x < 100; x++){
                     Queue[x] = "";
                        }
                     QueueHead = -1;
                     QueueTail = -1;
                    NumberItems = 0;

           ReadData();
           Compress();
           System.out.println(NewString);
     }
```

VB.NET
```
Sub Main()
     NewString = ""
     For x = 0 To 99
       Queue(x) = ""
     Next
     QueueHead = -1
     QueueTail = -1
     NumberItems = 0

     ReadData()
     Compress()
     Console.WriteLine(NewString)
     Console.ReadLine()
 End Sub
```

2(f)(i)

Python
```
NewString = ""
Queue = []
for x in range(100):
        Queue.append("")
QueueHead = -1
QueueTail = -1
NumberItems = 0
ReadData()
Compress()
print(NewString)
```

2(f)(ii)

1 mark for screenshot showing output

Example

```
19051306
```


6(a)

1 mark for identification **1 mark** for matching description

e.g.
- ASCII
- 7/8 bits per character // represents 128/256 characters // represents all characters from Latin alphabet

- UNICODE
- 8/16/32 bits per character // represents 256/65536+ characters // represents all characters in all languages

6(b)(i)

1 mark for:

256 // 2^8

6(b)(ii)

1 mark for:

Increased file size

6(b)(iii)

1 mark for each bullet point (**max 2**)

e.g.
- The change may not be noticeable // Data removed is usually not noticed by the human eye
- … for example, changes in shade/detail
- It produces a larger decrease in file size compared to lossless // Lossy decreases file size considerably

6(c)(i)

1 mark for each bullet point (**max 2**)

- Value/magnitude/size of the analogue sound wave is measured a set number of times each second/time / at set intervals
- Each sample/reading/measurement is given the binary number and stored in sequence

6(c)(ii)

1 mark for each correct point and **1 mark** for matching expansion

e.g.
- Decrease sample rate
- … fewer samples/readings/measurements stored per second // fewer bits per second stored

- Decrease sample resolution
- … fewer bits per sample/reading/measurement // each sample has fewer bits

- Sound outside of set/human hearing range is removed
- … fewer measurements are stored / decreases the number of possible binary values so fewer bits are stored

7(a)(i)	**1 mark** for each correct definition Colour depth: • the number of bits used to represent a colour // the number of colours that can be represented in an image File header: • stores data about the image file / metadata
7(a)(ii)	**1 mark** for each correct explanation Image quality: • Decreasing resolution means details within the image are lost because there are fewer pixels // Increasing resolution means the image is more detailed because there are more pixels File size: • Decreasing the resolution will decrease the file size because there are fewer pixels therefore less data // Increasing the resolution will increase the file size because there are more pixels therefore more data
7(a)(iii)	**1 mark** for correct method For example: • Run-Length Encoding
7(b)	**1 mark** for each correct definition Property: • an attribute of a drawing object // data about a shape // defines one aspect of the appearance of a drawing object Drawing list: • all the drawing objects/shapes in an image // stores the commands/descriptions / mathematical equations required to draw each object