1(a)(i)	B3A
1(a)(ii)	0001 0000 1000
1(a)(iii)	**1 mark** for the working **1 mark** for the correct denary value **Working:** Method 1: 1111 1011 1100 Flip the bits 0000 0100 0011 Add 1 0000 0000 0001 + 0000 0100 0100 Method 2: -2048 + 1024 + 512 + 256 + 128 + 32 + 16 + 8 + 4 // -128 + 32 + 16 + 8 + 4 **Denary value:** -68
1(b)(i)	(1) 00101011
1(b)(ii)	**1 mark** per bullet point, **max 2 marks** • An overflow error occurs • The answer cannot be represented in the number of bits available // The answer is larger than the maximum positive number that can be stored in the register // The answer is smaller than the most negative number that can be stored in the register

<table>
<tr><td>7(a)</td><td>1 mark for each correct description, max 2 marks

<table>
<tr><th>Type of IP address</th><th>Description</th></tr>
<tr><td>Static</td><td>IP address that does not change each time a device connects to the network</td></tr>
<tr><td>Public</td><td>IP address assigned to a device on a network to allow the device to be visible on the internet</td></tr>
</table>
</td></tr>
<tr><td>7(b)</td><td>1 mark per bullet point for justification, max 2 marks

Choice: Invalid (no mark for the choice)

Justification:

• It is not IPv4 because the maximum denary value should be 255, this uses 256

• It is not IPv6 because it uses a full stop to separate sections and not colon // It is not IPv6 because there are not enough groups</td></tr>
<tr><td>7(c)</td><td>1 mark per bullet point, max 4 marks

• The browser checks its cache for the URL
• The browser parses the URL and splits it into its component parts
• The browser finds the IP address for the domain name by querying a DNS server
• The browser receives the matching IP address from the DNS server
• The browser creates connection with the web server with the matching IP address
• A request for the resource is sent to the web server with that IP address
• The browser renders and displays the result
• The IP address is stored in the browser cache for future use</td></tr>
<tr><td>7(d)(i)</td><td>1 mark for the working
1 mark for the correct answer

Working:
(512 * 2048 * 8) / (8 * 1024) // (512 * 2048) / 1024
// (2^9 * 2^{11} * 2^3) / (2^3 * 2^{10}) // (2^9 * 2^{11}) / 2^{10}

Answer: 1024 kibibytes // 2^{10} kibibytes</td></tr>
<tr><td>7(d)(ii)</td><td>1 mark per bullet point, max 2 marks

• The file will contain metadata
• The file will have a header</td></tr>
<tr><td>7(d)(iii)</td><td>1 mark per bullet point, max 2 marks

• Sequences of consecutive identical colours / pixels
• ... are stored as the colour value and the number of times it occurs consecutively</td></tr>
</table>

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
```