1 (a) (i) Convert the binary number into hexadecimal.

101100111010

..... [1]

(ii) Convert the denary number into Binary Coded Decimal (BCD).

108

..... [1]

(iii) Convert the 12-bit two's complement binary integer into denary.

Show your working.

111110111100

Working .....

.....

.....

.....

Denary value .....

[2]

(b) (i) The following binary addition is performed using 8-bit registers.

Complete the calculation using binary addition.

```
  1 0 1 1 0 0 1 1
+ 0 1 1 1 1 0 0 0
```

[1]

(ii) Name and describe the error that can occur when binary addition is performed.

.....

.....

.....

..... [2]

7 Web browsers use Internet Protocol (IP) addresses.

(a) Complete the table by describing the following types of IP address.

Type of IP address	Description
Static	....
Public	....

[2]

(b) Consider the following IP address:

256.0.0.A

Circle whether this IP address is IPv4, IPv6 or an invalid IP address.

IPv4 IPv6 Invalid

Justify your choice.

.....

.....

.....

..... [2]

(c) A user types a Uniform Resource Locator (URL) into the address bar of a web browser to access a web page.

Explain how the web browser uses the URL to access the web page.

.....

.....

.....

.....

.....

.....

.....

[4]

(d) A bitmap file is downloaded. The image has a maximum of 256 colours and measures 512 pixels wide by 2048 pixels high.

(i) Calculate an estimate of the file size of the bitmap in kibibytes.

Show your working.

Working .....

.....

.....

.....

File size of bitmap ..... kibibytes

[2]

(ii) Explain why the actual file size may be larger than the one calculated in **(d)(i)**.

.....

.....

.....

..... [2]

(iii) Explain how a bitmap image is compressed using run-length encoding (RLE).

.....

.....

.....

..... [2]

2 A program reads string data from a text file into a linear queue and then compresses the data.

The queue is created using the global 1D array `Queue`. The queue can store up to 100 elements. Each element is initialised to the empty string `""`

The queue has two pointers that are global to the program:

- `QueueHead` that points to the index of the first element in the queue, initialised to `-1`
- `QueueTail` that points to the index of the last element in the queue, initialised to `-1`

The global variable `NumberItems` stores the number of elements stored in the queue, initialised to `0`

(a) Write program code to declare and initialise `Queue`, `QueueHead`, `QueueTail` and `NumberItems`

Save your program as **Question2_N25**.

Copy and paste the program code into part **2(a)** in the evidence document.

[2]

(b) The function `Enqueue()` takes a string as a parameter. The function checks if the queue is full, and returns Boolean `FALSE` if the queue is full.

If the queue is not full, the parameter is inserted into the next position in `Queue`. The function updates the appropriate pointer(s), updates `NumberItems` and then returns Boolean `TRUE`

Write program code for `Enqueue()`

Save your program.

Copy and paste the program code into part **2(b)** in the evidence document.

[5]

(c) The function `Dequeue()` returns the string `"False"` if the queue is empty.

If the queue is not empty, the function returns the next element in the queue. The function updates the appropriate pointer(s) and updates `NumberItems`

Write program code for `Dequeue()`

Save your program.

Copy and paste the program code into part **2(c)** in the evidence document.

[3]

(d) The text file `BinaryData.txt` stores individual binary digits, `'1'` and `'0'`. Each digit is on a new line. For example, the first line in the text file stores `'1'`, the second line stores `'1'`

The procedure `ReadData()` reads in each line from the text file `BinaryData.txt` and inserts it into the queue using the appropriate method.

The procedure needs to work for a text file with any number of lines up to a maximum of 100.

Write program code for `ReadData()`

Save your program.

Copy and paste the program code into part **2(d)** in the evidence document.

[5]

(e) The string data in the text file is compressed.

The compression algorithm counts the number of times each binary digit appears consecutively, then stores the binary digit followed by the number of times it appears. The algorithm stores the compressed data in a single string.

For example, if the text file contains the data:

```
1
1
0
0
0
1
1
1
```

The compression algorithm will create the string `"120313"` because there are two `'1'` digits, followed by three `'0'` digits, followed by three `'1'` digits.

The procedure `Compress()` uses `Dequeue()` to remove each element from the queue in turn. The procedure then compresses the data following the compression algorithm described. The new compressed string is stored in the global variable `NewString`

You can assume that one binary digit will never appear more than nine times consecutively in the sequence.

You can assume that there will always be at least one item in the queue.

Write program code for `Compress()`

Save your program.

Copy and paste the program code into part **2(e)** in the evidence document.

[6]

(f) **(i)** Write program code for the main program to:

- call `ReadData()`
- call `Compress()`
- output the content of the compressed string.

Save your program.

Copy and paste the program code into part **2(f)(i)** in the evidence document.

[2]

(ii) Test your program.

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot(s) into part **2(f)(ii)** in the evidence document.

[1]