6 **(a)** A sound file is compressed by reducing the sampling rate.

State whether this is lossless or lossy compression. Justify your choice.

Type of compression .....

Justification .....

.....

.....

.....

[1]

(b) The following table shows some words and corresponding denary values.

Word	Denary value
Computing	55
Science	56
Computers	57
are	58
Brilliant!	59
is	60
Fun!	61
Amazing!	62

The following table shows three bytes of data that have been received.

Use the table to find the corresponding words from the binary values received.

Binary value	00111000	00111100	00111110
Word			

Working .....

.....

.....

.....

[1]

(c) A computer system uses even parity. The least significant (rightmost) bit of each byte is the parity bit.

(i) Complete the byte by writing the missing parity bit:

parity bit ↓

0	1	0	1	1	1	0	

[1]

(ii) The computer also uses parity block check. The parity block check uses even parity. Computer A transmits four bytes of data to computer B, followed by a parity byte. Computer B receives the following sequence of bytes.

parity bit ↓

	1	0	1	1	0	1	1	1
	0	1	1	1	0	0	0	0
	0	0	0	1	1	0	1	1
	0	1	1	1	0	1	0	0
parity byte →	1	0	1	0	0	0	0	0

Following transmission, one of the four bytes of data has an error in one of the bits.

Circle the bit that has been altered during the data transfer. [1]

(d) A bitmap image has a resolution of 1000 pixels wide by 2000 pixels high. The colour depth is 16 bits.

Calculate an estimate of the file size in megabytes.

Show your working.

.....

.....

.....

.....

.....

.....

File size ..... megabytes

[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]