0 1 Run length encoding (RLE) is a form of compression that creates frequency/data pairs to describe the original data.

For example, an RLE of the bit pattern `00000011101111` could be `6 0 3 1 1 0 4 1` because there are six `0`s followed by three `1`s followed by one `0` and finally four `1`s.

The algorithm in **Figure 7** is designed to output an RLE for a bit pattern that has been entered by the user.

Five parts of the code labelled **L1**, **L2**, **L3**, **L4** and **L5** are missing.

- Note that indexing starts at zero.

Figure 7

```
pattern ← L1
i ← L2
count ← 1
WHILE i < LEN(pattern)-1
   IF pattern[i] L3 pattern[i+1] THEN
      count ← count + 1
   ELSE
      L4
      OUTPUT pattern[i]
      count ← 1
   ENDIF
   L5
ENDWHILE
OUTPUT count
OUTPUT pattern[i]
```

0 1 . 1 Shade **one** lozenge to show what code should be written at point **L1** of the algorithm.

[1 mark]

A `OUTPUT` ⬭

B `'RLE'` ⬭

C `True` ⬭

D `USERINPUT` ⬭

0 1 . 2 Shade **one** lozenge to show what value should be written at point **L2** of the algorithm.

[1 mark]

A -1 ⬭

B 0 ⬭

C 1 ⬭

D 2 ⬭

0 1 . 3 Shade **one** lozenge to show what operator should be written at point **L3** of the algorithm.

[1 mark]

A = ⬭

B ≤ ⬭

C < ⬭

D ≠ ⬭

0 1 . 4 Shade **one** lozenge to show what code should be written at point **L4** of the algorithm.

[1 mark]

A `count` ⬭

B `count ← count - 1` ⬭

C `count ← USERINPUT` ⬭

D `OUTPUT count` ⬭

0 1 . 5 Shade **one** lozenge to show what code should be written at point **L5** of the algorithm.

[1 mark]

A `i ← i * 2` ⬭

B `i ← i + 1` ⬭

C `i ← i + 2` ⬭

D `i ← i DIV 2` ⬭

0 1 . 6 State a run length encoding of the series of characters `ttjjeeess`

[2 marks]

0 1 . 7 A developer implements the algorithm shown in **Figure 7** and tests their code to check that it is working correctly. The developer tests it only with the input bit pattern that consists of six zeros and it correctly outputs `6 0`.

Using example test data, state **three** further tests that the developer could use to improve the testing of their code.

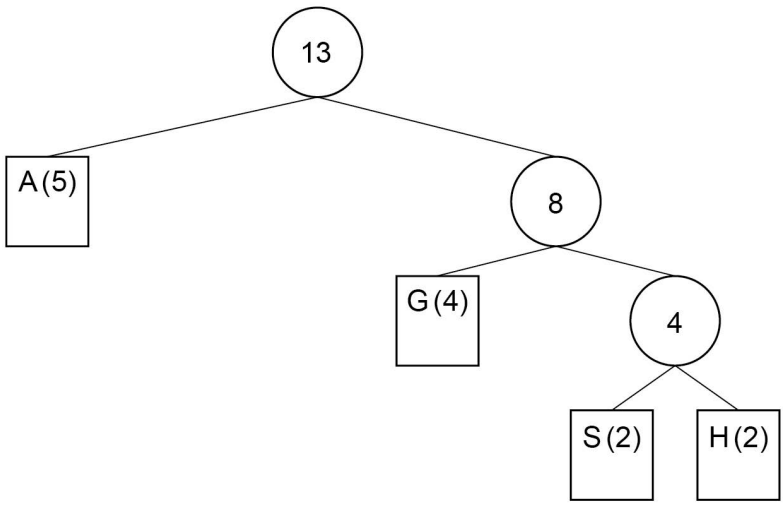
[3 marks]

0 2 The Huffman tree shown in **Figure 6** was created to encode the string shown in **Figure 5**. The frequency of each character is shown in brackets. For example, the letter A appears five times within the string shown in **Figure 5**.

Figure 5

AAGHHGGSAAASG

Figure 6



0 2 . 1 Complete the code table below for characters G, S and H for the Huffman tree shown in **Figure 6**. The code for character A has already been completed.

[3 marks]

Character	Binary code
A	0
G	
S	
H	

0 2 . 2 The string shown in **Figure 5** could also be encoded using ASCII. ASCII uses 7 bits to represent each character.

How many bits are **saved** by using Huffman coding rather than ASCII to represent the string shown in **Figure 5**?

You **must** show your working.

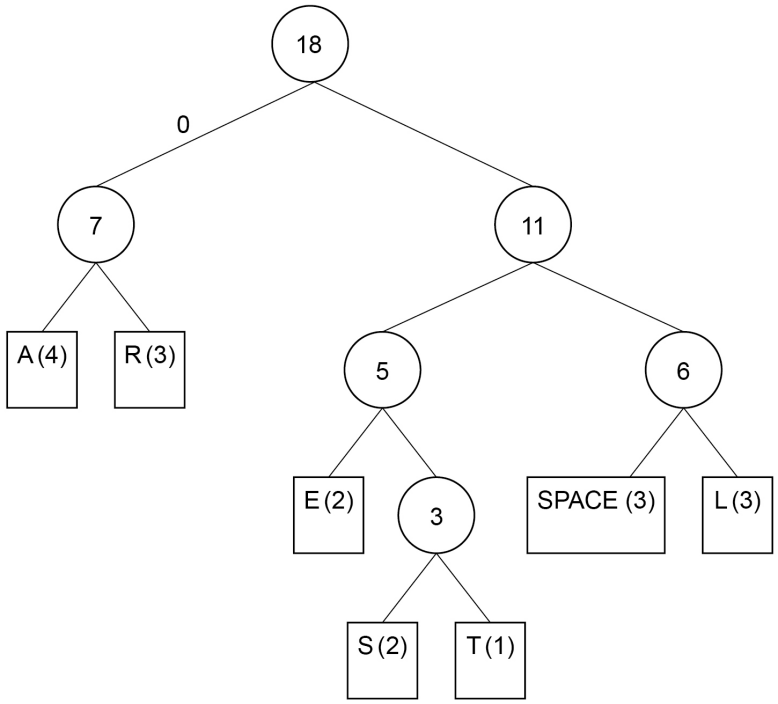
[4 marks]

Answer: _____

Turn over for the next question

0 3 The Huffman tree in **Figure 1** was generated for the string `ARE ALL STARS REAL`

Figure 1



0 3 . 1 Part of the string `ARE ALL STARS REAL` was incorrectly encoded as in **Figure 2** below.

Figure 2

`111100001010101 1`

What string does this encoding represent?

[1 mark]

0 3 . 2 What would be the correct binary encoding for the substring `STAR`?

Write the correct encoding below the letters in the table.

[2 marks]

S	T	A	R

0 4 . 1 Shade **one** lozenge to show which statement best describes data compression.

[1 mark]

A The process of calculating the file size of a saved file. ⬭

B The process of encoding characters into more than one language. ⬭

C The process of encoding information to try and use fewer bits than the original. ⬭

D The process of removing necessary data from a file. ⬭

0 4 . 2 Give **two** reasons why data compression is often used.

[2 marks]

1 _____

_____

2 _____

_____

Run length encoding (RLE) is one method of compressing data.

0 4 . 3 State the feature of data that allows it to be compressed effectively using RLE.

[1 mark]

_____

0 4 . 4 Describe how RLE works. In your answer you **must** use an example.

[2 marks]

_____

_____

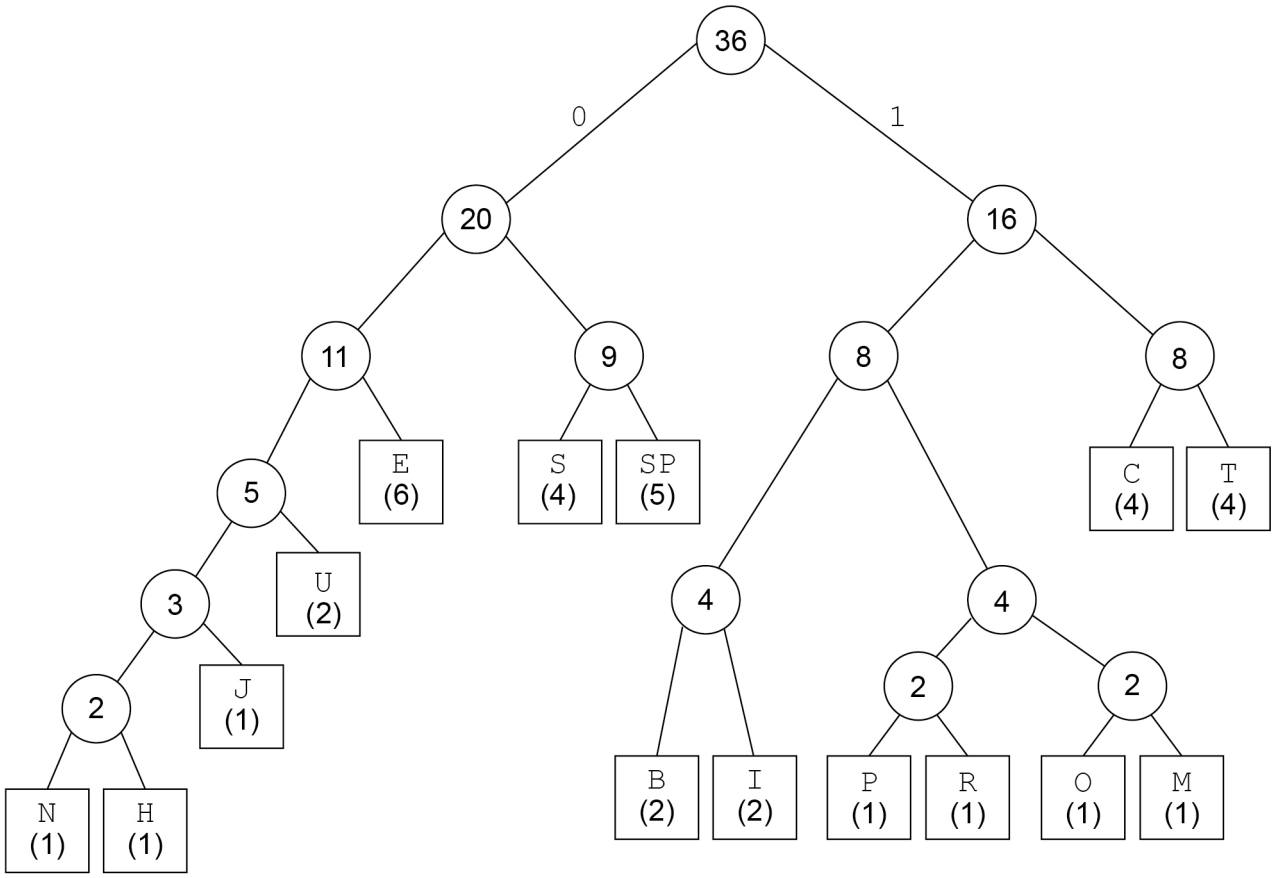
_____

_____

Turn over for the next question

0 5 **Figure 1** shows a Huffman tree that has been created to represent the string shown in **Figure 2**.

Figure 1



SP represents a space character

Figure 2

COMPUTER SCIENCE IS THE BEST SUBJECT

0 5 . 1 Use the Huffman tree in **Figure 1** to state the Huffman encoding for the string MOST

[3 marks]

M	O	S	T

0 5 . 2 A student was asked to describe how a Huffman tree could be created for the string in **Figure 2**. Her response was:

"I would count the number of times each character appears in the string and create a frequency table sorted alphabetically. For example, the letter S has the highest frequency in **Figure 2**. Next I would take the two characters with the largest frequencies and combine them into a new node. The new node would be added to the end of the frequency table. The two characters with the lowest remaining frequencies are now combined into a new node and the process is repeated until all the characters have been added to nodes and the tree created."

State **four** mistakes the student has made in her response.

[4 marks]

1 _____

_____

2 _____

_____

3 _____

_____

4 _____

_____

0 5 . 3 When the Huffman tree in **Figure 1** is used, the string in **Figure 2** can be represented using 130 bits.

The 36-character string shown in **Figure 2** could also be encoded using ASCII.

How many bits are **saved** when Huffman coding is used rather than ASCII to represent the string shown in **Figure 2**?

You **must** show your working.

[2 marks]

_____

_____

_____

_____

Answer _____

0 6 . 1 State **two** reasons why data are compressed.

[2 marks]

1 _____

_____

2 _____

_____

0 6 . 2 **Figure 2** shows a string.

Figure 2

MISSISSIPPI

One method for compressing data is run length encoding (RLE).

When using RLE, the data in **Figure 2** become:

1M 1I 2S 1I 2S 1I 2P 1I

Explain why RLE is **not** a suitable method for compressing the data in **Figure 2**.

[2 marks]

_____

_____

_____

_____

_____

0 6 . 3 Another method for compressing data is Huffman coding. In Huffman coding, the codes for the characters can be created based on their position in a tree.

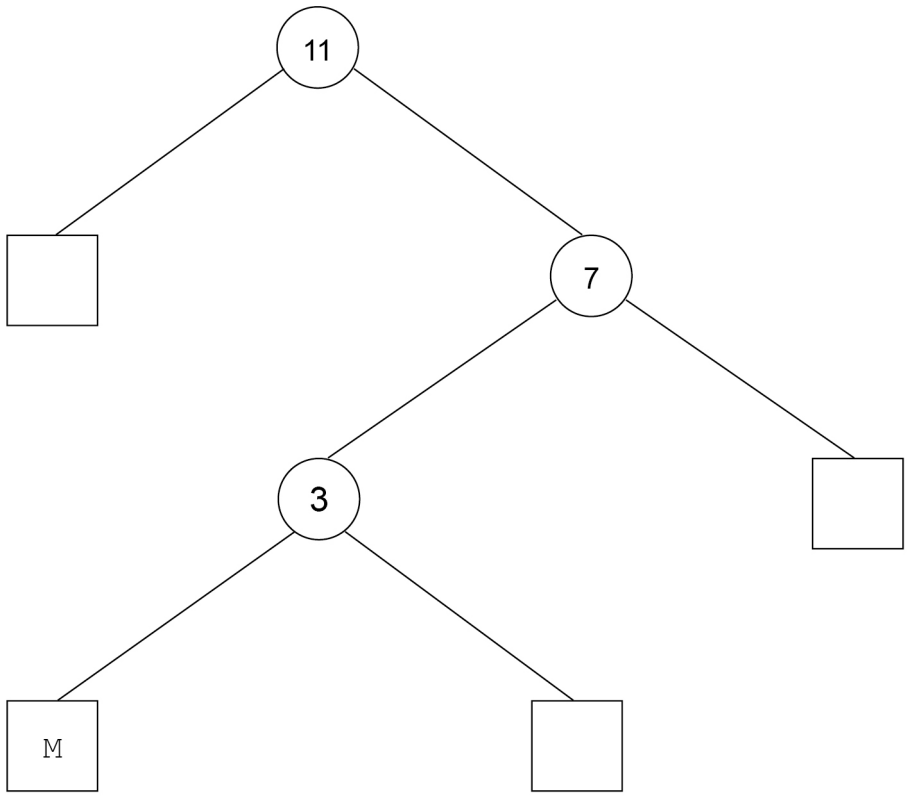
Figure 3 shows a Huffman code for each different character in the string in **Figure 2**.

Figure 3

Character	Binary code
M	100
I	0
S	11
P	101

Complete the Huffman tree below to show the position of the characters I, S and P using the codes from **Figure 3**.

[1 mark]



0 7 . 4 A black and white image has been compressed using run length encoding (RLE).

The first bit in each byte of the bit pattern represents the colour and the remaining seven bits of the byte represent the number of pixels in the run.

The image has a run of 60 black pixels followed by a run of 30 white pixels and is represented by the bit pattern shown in **Figure 2**.

Figure 2

0	0	1	1	1	1	0	0	1	0	0	1	1	1	1	0

Using the same RLE method, give the bit pattern for a black and white image that has a run of 64 white pixels followed by a run of 15 black pixels.

Write your answer in **Table 1**.

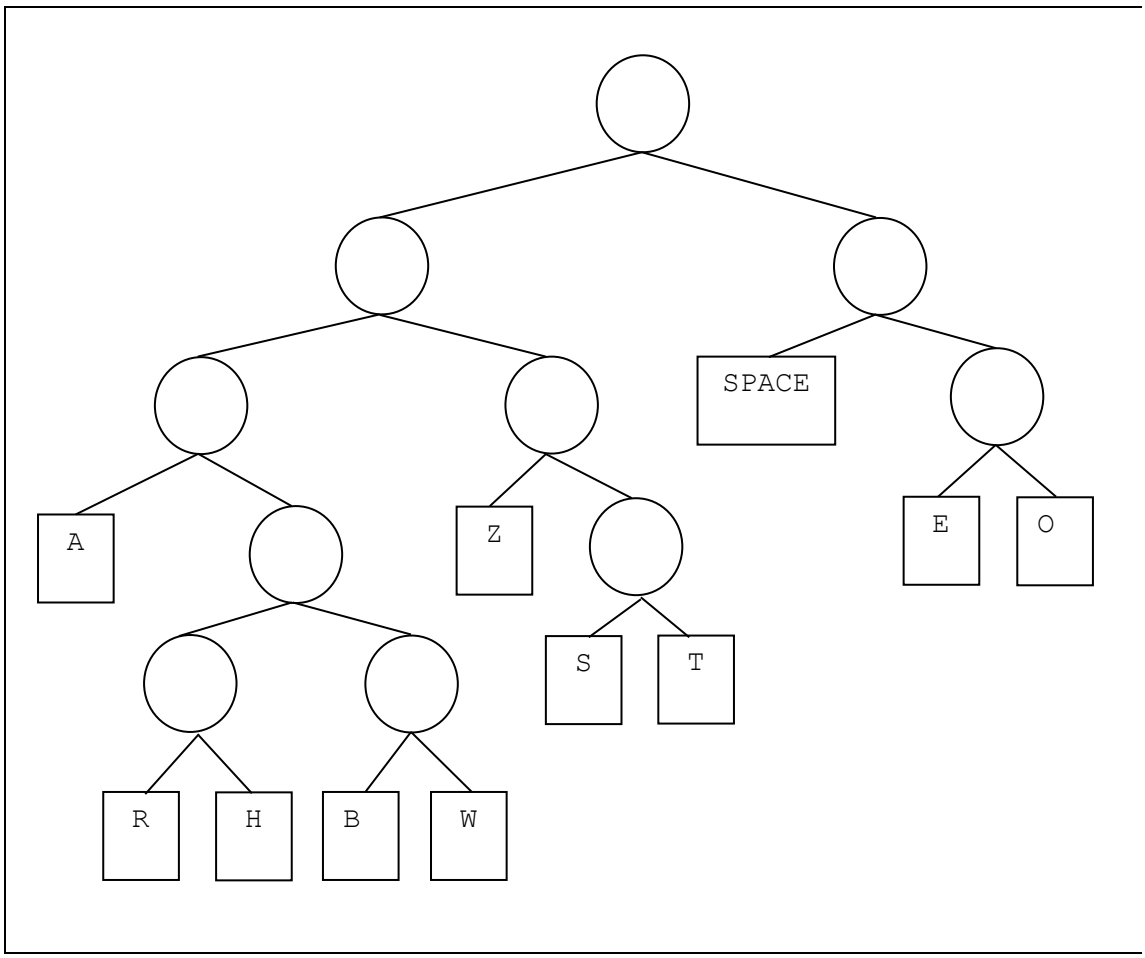
[2 marks]

Table 1

When data is stored in a computer it is often compressed. One method that can be used to compress text data is Huffman coding. To produce a Huffman code each character in a piece of text is placed in a tree, with its position in the tree determined by how often the character was used in the piece of text.

A Huffman tree for the text `ZOE SAW A ZEBRA AT THE ZOO` is shown in **Figure 3**.

Figure 3



Using this Huffman tree, the Huffman coding for the character E would be the bit pattern 110 because from the top of the tree E is to the right, then right again and then left.

The character Z is represented by the bit pattern 010 because from the top of the tree Z is to the left, then right and then left.

0 8 . 7 Using the Huffman code in **Figure 3**, complete the table to show the Huffman coding for the characters O, SPACE and B. **[3 marks]**

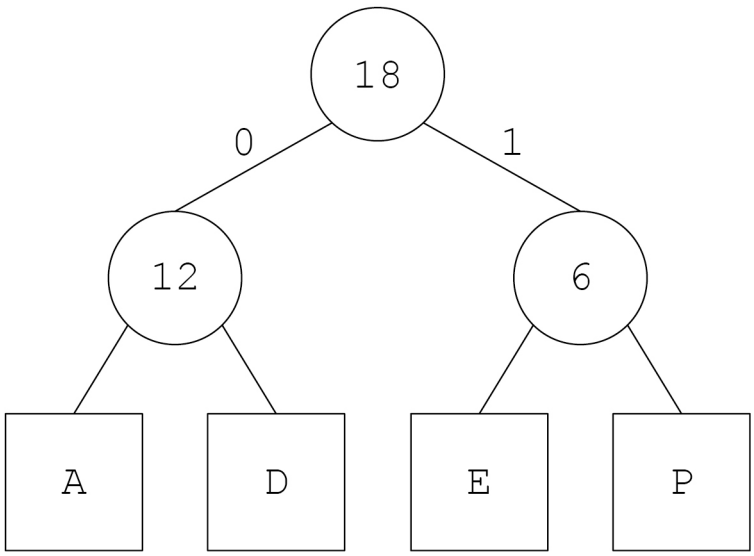
Character	Huffman coding
O	
SPACE	
B	

0 8 . 8 Using Huffman coding, the text ZOE SAW A ZEBRA AT THE ZOO can be stored in 83 bits.

Calculate how many additional bits are needed to store the same piece of text using ASCII. Show your working. **[3 marks]**

0 9 . 1 **Figure 7** contains a Huffman tree.

Figure 7



The Huffman tree in **Figure 7** was used to encode a string, which resulted in the following bit pattern:

0001011001

State the string that this bit pattern represents.

[2 marks]

0 9 . 2 **Table 1** shows the Huffman codes for the characters used in the string HESELLSSEASHELLASHES

Table 1

Character	Character frequency	Huffman code
S	6	11
E	5	10
L	4	00
H	3	011
A	2	010
	20	

Calculate how many bits would be saved if the string HESELLSSEASHELLASHES was encoded using the Huffman codes shown in **Table 1**, rather than using ASCII.

You should show your working.

[3 marks]

Number of bits saved _____