UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B.Sc. Examination 2011

Computing

IS53010A Data Compression

Duration: 2 hours and 15 minutes

Date and time:

There are five questions in this paper. You should answer no more than three questions. Full marks will be awarded for complete answers to a total of three questions. Each question carries 25 marks. The marks for each part of a question are indicated at the end of the part in [.] brackets.

There are 75 marks available on this paper.

Electronic calculators must not be programmed prior to the examination. Calculators which display graphics, text or algebraic equations are not allowed.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Question 1

(a) Consider the following statement about the limit of lossless compression. Outline your view about the truth of the statement and justify your answer. [5]

"More than 99% of files cannot be compressed even by one byte under lossless compression schemes."

(b) Explain the concept of *temple redundancy* with the aid of an example in diagram. [5]

(c) Consider a source of binary alphabet (A, B) with a probability distribution (p_A, p_B). Under what probability distribution will the static Huffman coding algorithm achieve an optimal performance? Under what probability distribution will the algorithm give the worst performance? Give your reasons. [8]

(d) Prefix codes have important characteristics. Give an example of a prefix code of length 4. Discuss the advantages of using a prefix code. Demonstrate, with the aid of an example, how to decide that a binary code is not uniquely decodable. [7]

Question 2

Consider the 4×4 matrix $\mathbf{A}$ below that represents the pixel values of a partial image.

$$\mathbf{A} = \begin{array}{|c|c|c|c|}\hline 4 & 8 & 4 & 8 \\ \hline 1 & 2 & 4 & 6 \\ \hline 8 & 4 & 5 & 5 \\ \hline 2 & 4 & 8 & 5 \\ \hline \end{array}$$

(a) Derive the residual matrix $\mathbf{R}$, applying the JPEG predictive rule $x = Q + (S - T)/2$, where x is defined using the schema $\begin{array}{|c|c|}\hline \text{T} & \text{S} \\ \hline \text{Q} & \text{x?} \\ \hline \end{array}$. Each matrix entry should be *rounded* to the nearest integer. [5]

(b) Derive a canonical minimum variance static Huffman code, for (i) the partial image matrix $\mathbf{A}$, and (ii) the residual matrix $\mathbf{R}$. [10]

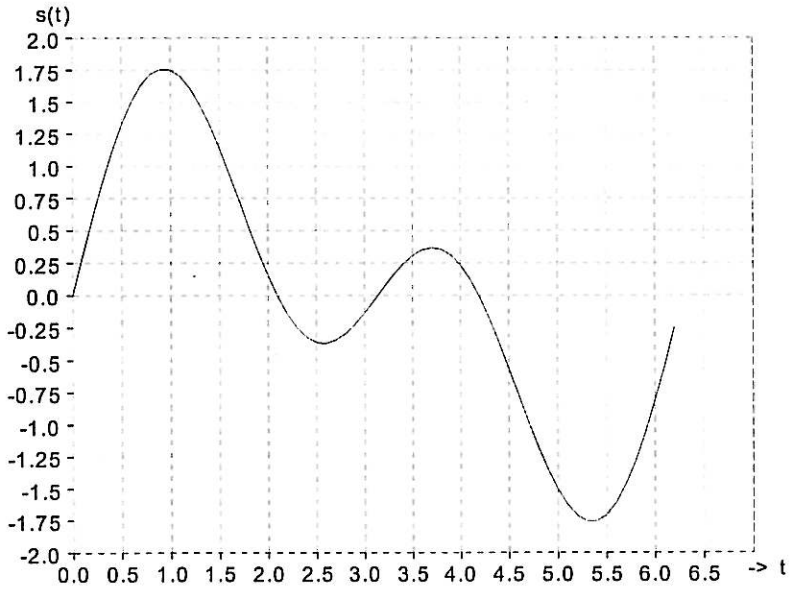
(c) Compare the two compression approaches above in terms of code efficiency $E = H/\bar{l}$, where H is the first order entropy and $\bar{l}$ refers to the average length of the code. Show all your work before summarising the results for $\mathbf{A}$ and $\mathbf{R}$ in a table as follows: [10]

	H	$\bar{l}$	E
A			
R			

Question 3

(a) Explain the main efficiency problem of the canonical minimum-variant Huffman encoding algorithm and how the problem can be solved by maintaining two lists. [4]

(b) Consider the signal $s(t) = sin(t) + sin(2t)$ for $t \in [0, 2\pi]$ in the diagram below.



i. What is the minimum sample rate for a valid sampling? Justify your answer. [3]

ii. Demonstrate the sample data taken at $t = 1, 2, 3, 4, 5, 6$. Assume the precision of two decimals. [2]

iii. Conduct a simple quantisation on the sample data taken previously, rounding the data to nearest integers. [2]

iv. Demonstrate how to represent the first two positive sample integers by a Grey Code of length 3. [2]

v. Demonstrate how the HDC run-length algorithm can be applied to the sample data after the quantisation. Outline the HDC run-length algorithm and explain the meaning of control symbols in the algorithm. [4]

(c) Following the approach of the LZW algorithm, decode the tokens (1 1 2 259 5 2) step by step. Assume that the dictionary initially contains single characters A–Z and occupies cells at 1–256 only. [8]

Question 4

(a) Draw a flow chart to outline the general stages for developing a compression algorithm. [5]

(b) What is the main difference between a *lossless* compression and *lossy* compression? What does a lossy compression usually aim to do? Give an example of real life data that is suitable for lossy compression. [5]

(c) Consider a source (A, B, C, D) with a probability distribution (0.4, 0.3, 0.2, 0.1). Discuss whether each of the following statements is true or false. Provide supporting arguments or evidence to justify your answer. [6]

i. The binary tree representing a fixed length binary code is not optimal.
ii. The Shannon-Fano code for the given source in the question is optimal.

(d) Explain and demonstrate how the compression efficiency of the Shannon-Fano encoding can be improved by alphabet extension. Use a binary alphabet (A, B) with the probability of A being 0.3 as an example. [9]

Question 5

(a) Consider the alphabet (A, B, C, D) of a source. Discuss the possibility of finding: [5]

i. A uniquely decodable binary code in which the codeword for A is of length 2, that for B of length 1 and for both C and D of length 3.

ii. A shorter variable length prefix code than the one described in (a)i.

Provide evidence or justification for your answers.

(b) Draw a flowchart to outline the adaptive Huffman algorithm for encoding. [5]

(c) Draw a diagram to outline the LZ77 decoding algorithm. [10]

Following the approach of the LZ77 algorithm, decode the tokens `(0,ASCII(A))`, `(0,ASCII(A))`, `(0,ASCII(B))`, `(0,ASCII(A))`, `(0,ASCII(C))`, `(0,ASCII(C))`, `(5,2)`, `(6,2)`, `(0,ASCII(A))`, `(8,3)`, `(0,ASCII(C))`. Assume that the length of the history buffer is $H = 8$ and of the lookahead buffer is $L = 6$. The history buffer is empty initially.

(d) Explain what each of the variables *L, x, s1, p2* represents in the segment of the Arithmetic decoding algorithm below. Demonstrate how the algorithm works with the aid of a small example. Assume a source (A, B) with a probability distribution (0.2, 0.8).

Hint: You may trace variable values at the end of each iteration 0–5 for an input 0.43 as suggested in the table below. Use iteration 0 to describe the initial state and add necessary assumptions for a specific source. Finally, derive the decoded text. [5]

```
1. L <- 0 and d <- 1
2. If x is within [L, L+d*p1)
     then output s1, leave L unchanged, and
          set d <- d*p1
     else if x is within [L+d*p1, L+d)
          then output s2, set L <- L+d*p1 and d <- d*p2
3. If (the_number_of_decoded_symbols
          < the_required_number_of_symbols)
     then go to step 2.
```

Iteration	L	d	d*p1	d*p2	[L, L+d*p1)	[L+d*p1, L+d)	Output
0							
1							
2							
3							
4							
5							