THE UNIVERSITY OF AUCKLAND

Department of Computer Science
COMPSCI 314 S1 C 2004 : Assignment 3
Due Tuesday April 27 2004, 4.00 pm.

Q1. (i) Generate the codeword *c(x)* for the message *i(x)* below, forming the CRC using the generator polynomial $g(x) = x^4 + 1$.

i(x) = 1001010110100110 **[3 marks]**

All divisions operations show only steps where the subtraction "succeeds" (quotient = 1)

```
                1 0 0 1 1 1 0 0 0 1 1 0 0 0 0 0
               ---------------------------------------
    1 0 0 0 1 ) 1 0 0 1 0 1 0 1 1 0 1 0 0 1 1 0 0 0 0 0
              - 1 0 0 0 1
                ----------
                  0 0 1 1 1
                    - 1 0 0 0 1
                      ----------
                        1 1 0 0 1
                      - 1 0 0 0 1
                        ----------
                          1 0 0 0 0
                        - 1 0 0 0 1
                          ----------
                            0 0 0 1 1
                                - 1 0 0 0 1
                                  ----------
                                    1 0 0 0 1
                                  - 1 0 0 0 1
                                    ----------
                                      0 0 0 0 0
                                               =========
                                   Remainder =  0 0 0 0

  Codeword =   1 0 0 1 0 1 0 1 1 0 1 0 0 1 1 0  0 0 0 0
```

(ii) Show that the received message $v(x) = c(x)$ gives a zero syndrome. **[2 marks]**

```
                1 0 0 1 1 1 0 0 0 1 1 0 0 0 0 0
               ---------------------------------------
    1 0 0 0 1 ) 1 0 0 1 0 1 0 1 1 0 1 0 0 1 1 0 0 0 0 0
              - 1 0 0 0 1
                ----------
                  0 0 1 1 1
                    - 1 0 0 0 1
                      ----------
                        1 1 0 0 1
                      - 1 0 0 0 1
                        ----------
                          1 0 0 0 0
                        - 1 0 0 0 1
                          ----------
                            0 0 0 1 1
                                - 1 0 0 0 1
                                  ----------
                                    1 0 0 0 1
                                  - 1 0 0 0 1
                                    ----------
                                      0 0 0 0 0
                                               =========
                                   Remainder =  0 0 0 0
```

(iii) Show that the received message $v(x) = c(x) + e(x)$, where $e(x) = x^7$ gives a detectable error. **[2 marks]**

```
                1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0
              ---------------------------------------
1 0 0 0 1 ) 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 0 0 0 0
            - 1 0 0 0 1
              -----------                A  the bad bit
                0 0 1 1 1
                  - 1 0 0 0 1
                    -----------
                      1 1 0 0 1
                    - 1 0 0 0 1
                      -----------
                        1 0 0 0 0
                      - 1 0 0 0 1
                        -----------
                          0 0 0 1 1
                              - 1 0 0 0 1
                                -----------
                                  1 0 1 0 1
                                - 1 0 0 0 1
                                  -----------
                                    0 1 0 0 0
                                  - 1 0 0 0 1
                                    -----------
                                      0 0 0 1 0
                                      =========
                                Remainder =  1 0 0 0
```

This whole answer has been corrected

(iv) Show that the check bits added by the division are the same as those generated by taking the *longitudinal* parity of the message, considered as a sequence of 4-bit "nibbles". $i(x)$ = 1001 0101 1010 0110 **[2 marks]**

Write the nibbles one above the other and then calculate the parity for each column —

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

Each column has 2 bits, and the added parity bits for even parity are all 0

(v) (Bonus) Explain why parts (i) and (iv) give the same check bits. **{2 marks}**

During the scan along $i(x)$, the generator is subtracted whenever the most significant bit of the partial remainder (or residue etc) is a 1. This subtraction exclusive-ORs a 1 with the bit 4 places to the right, which is the corresponding bit of the next nibble. The separate "columns" of the nibbles do not interfere with each other and the final remainder is the bit-wise exclusive-OR of the bits of each nibble, as was obtained by the longitudinal parity.

Question total [9 marks (+2)**]**

2003 Ans **As the generator scans along i(x), it does a subtraction whenever the most significant bit of the residue (or partial remainder etc) is a 1. This subtraction exclusive-ORs a 1 with the residue bit 4 places to the right, which is the corresponding bit of the next nibble. The different "columns" of the nibbles do not interfere with each other and the final remainder is the bit-wise exclusive-OR of the bits of each nibble, as was obtained by the longitudinal parity (see Handout 3, slide 9).**

Q2. (i) Generate the Hamming codeword $c(x)$ for the information octet $i(x)$=01001011.

(This question assumes the least-significant bit to the left, with bits numbered from 1). **[2 marks]**

Bit numbers (parity in red)	1	2	3	4	5	6	7	8	9	10	11	12	count selected 1 bits	force even parity	force odd parity
Insert info bits			0		1	0	0		1	0	1	1			
checked by bit 1	X		X		X		X		X		X		3	1	0
checked by bit 2		X	X			X	X			X	X		1	1	0
checked by bit 4				X	X	X	X					X	2	0	1
checked by bit 8								X	X	X	X	X	3	1	0
Codeword (even)	1	1	0	0	1	0	0	1	1	0	1	1			
Codeword (odd)	0	0	0	1	1	0	0	0	1	0	1	1			

As we know that the parity bits must be in positions 1, 2, 4 and 8, the information bits are put into the remaining positions. Now count the bits selected by each of the parity "masks", and then set the parity bit in each case to so that the count under each mask is even. We can count only the information bits, assuming that the parity bits are, so far, all zero.

(ii) Show that the generated codeword decodes indicating no error **[2 marks]**

Bit numbers (parity in red)	1	2	3	4	5	6	7	8	9	10	11	12	selected 1 bits	syndrome bits
Codeword (even)	1	1	0	0	1	0	0	1	1	0	1	1		
checked by bit 1	X		X		X		X		X		X		4	0
checked by bit 2		X	X			X	X			X	X		2	0
checked by bit 4				X	X	X	X					X	2	0
checked by bit 8								X	X	X	X	X	4	0

Count the codeword bits (including parity) as selected by each of the masks. Each parity-check group selects an even number of bits, showing that there is no error.

(iii) Show that the Hamming decode will correctly show an error in bit 7 of the codeword.

[2 marks]

Bit numbers (parity in red)	1	2	3	4	5	6	7	8	9	10	11	12	selected 1 bits	syndrome bits
Codeword (even)	1	1	0	0	1	0	**1**	1	1	0	1	1		
checked by bit 1	X		X		X		X		X		X		5	1
checked by bit 2		X	X			X	X			X	X		3	1
checked by bit 4				X	X	X	X					X	3	1
checked by bit 8								X	X	X	X	X	4	0

Bit 7 is now changed from 0 to 1. Again count the bits selected within each parity group. Bits 1, 2 and 4 "fail" with an odd count. producing a 1 in that bit of the syndrome. Bit 8 still has an even count and produces a 0 syndrome bit. The syndrome is thus 1*1+1*2+1*4+0*8 = 7, showing that there is error in bit 7.

Question total [6 marks]

Q3. *The answers to this question should be presented in tables similar to those in the handouts. Assume that the dictionary is initialised to the 'ASCII' codes 0–255, to cover all possible byte values; you start adding words at index 256. You may refer to one of these initial entries by its letter rather than its numeric index, such position* A *rather than* $41_{16} = 65_{10}$.

(i) Apply LZW compression to the string `PARAPARAUMU` **[5 marks]**

Many students noted that there is a problem at the end of the string being compressed. This problem, of handling End-Of-File, is seldom discussed in compression texts. (Likewise few texts discuss the "recursive entry" problem.)
The best way of handling this (NOT needed in the answer) is to have a special End File code (perhaps dictionary 256, and start adding at 257) which is emitted at the end and forces proper "wrap-up" of the text. Here I add a "*" as the EOF code.

PARAPARAUMU*

Pass	Input	Test String	Emit	Make Entry	Comment
1	P	P			
2	A	PA	P	PA	
3	R	AR	A	AR	
4	A	RA	R	RA	
5	P	AP	A	AP	
6	A	PA			PA exists
7	R	PAR	PA(256)	PAR	
8	A	RA			RA exists
9	U	RAU	RA(258)	RAU	
10	M	U	U	UM	
11	U	MU	M	MU	
12	*	U*	U		* forces EOF processing

Dictionary

Index	Contents
0 – 255	all byte values
256	PA
257	AR
258	RA
259	AP
260	PAR
261	RAU
262	UM
263	MU

Deduct **1 mark if ANY error in test string**
1 mark (max 2) for each error in emitted code
1 mark (max 2) for each error in the dictionary (first added entry MUST start at 256)
Can also deduct 1 mark for untidy presentation
NO deduction if EndFile not included, or last character not processed

(ii) Recover the original text by decompressing the output from part (i) **[5 marks]**

Pass	Receive	Make Entry	Deliver	Comment
1	P	–	P	
2	A	PA(256)	A	
3	R	AR(257)	R	
4	A	RA(258)	A	
5	PA(256)	AP(259)	PA	
6	RA(258)	PAR(260)	RA	
7	U	RAU(261)	U	
8	M	UM(262)	M	
9	U	MU(263)	U	
10	*	–	U	force end-file processing

Dictionary

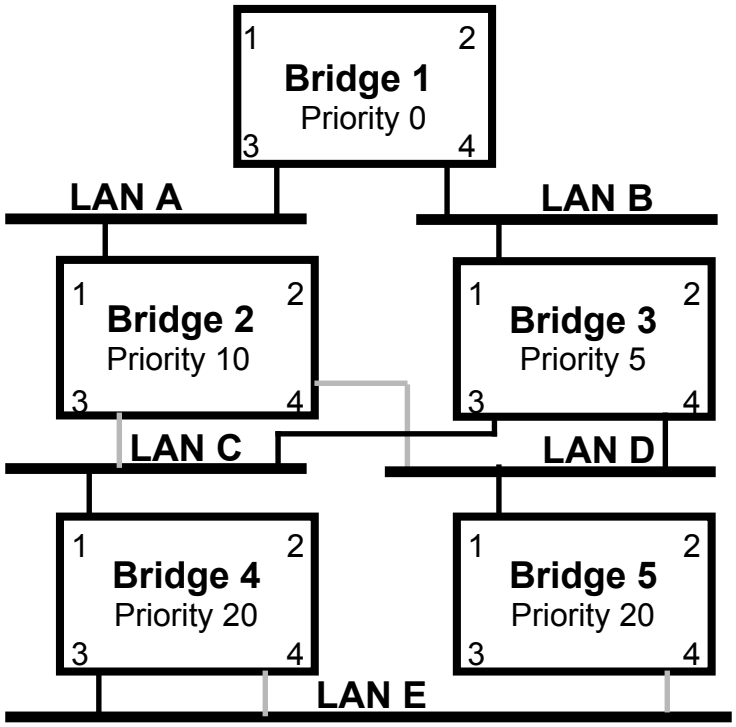
Index	Contents
0 – 255	all byte values
256	PA
257	AR
258	RA
259	AP
260	PAR
261	RAU
262	UM
263	MU

Deduct **1 mark (max 3) for each error in delivered code**
1 mark (max 2) for each error in the dictionary (MUST be same as in coder.)
1 mark if "received codeword" is not identical to "Emit" column of part (i)
NO deduction if EndFile not included, or last character not processed

Question total [10 marks]

Q4. Obtain the spanning tree for the network shown below.

- The notes had the BridgeID as only the MAC address, but here we use the full form with the Bridge ID being firstly a "bridge priority" (most significant) and then the Bridge Number (least significant).
  Bridges are therefore compared *first* on priority and *second* on Bridge number.
- All ports have the same cost, and each is identified by its "corner" of the bridge, such as Bridge 3, port 2 at the top-right corner might be written as port[3.2] (and is not used here).



Question total [5 marks]

Assume that each port has a cost of 1.

- The Root Bridge is clearly Bridge 1. [1 mark]
- LAN C connects through [Bridge 2 Port 3] and [Bridge 3 Port 5], both with cost of 1.
  But Bridge 3 has a smaller priority value and is preferred with its higher priority [1 mark]
- LAN D, by the same argument, also connects through Bridge 3. [1 mark]
- LAN E, has costs of 2 over all three connected ports, and all have the same priority.
  But Bridge 4 is preferred over Bridge 5 (lower number), [1 mark]and its port 3 is preferred over its port 4 (lower number), leaving [Bridge 4, Port 3] as the connection to LAN E [1 mark]. [2 marks]

The disabled connections are shown in grey in the diagram above.

Assignment TOTAL = 30 marks (+2)

Bit numbers (parity in red)	1	2	3	4	5	6	7	8	9	10	11	12	selected 1 bits		force even
parity									force	odd	parity				
Insert info bits			**0**			**0**			**1**			**1**			
													2	**0**	
						x	**x**						**0**	**0**	
												x	**2**	**0**	
checked by bit 8												**x**	**3**	**1**	
Final codeword		**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**		
											1	1			

Bit numbers (parity in red)	1	2	3	4	5	6	7	8	9	10	11	12	selected 1 bits		force even
parity									force	odd	parity				
Insert info bits			**0**			**0**			**1**			**1**			
													2	**0**	
						x	**x**						**0**	**0**	
												x	**2**	**0**	
checked by bit 8												**x**	**3**	**1**	
Final codeword		**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**		
											1	1			