1. (i) The three reconstruction levels are:
$s_1 = \frac{1}{2}\frac{255}{3} \cong 43$, $s_2 = 43 + 85 = 128$, $s_3 = 128 + 43 = 171$.

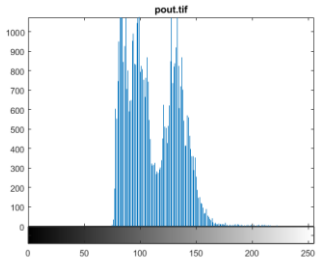
(ii) The equation which describes the pdf function is of the form $y = \frac{2}{255 \cdot 255}x$. Based on this we can find the probabilities of the 3 reconstruction levels.
The probability of s_1 is 1/9.
The probability of s_2 is 1/3. The probability of s_3 is 5/9.
Huffman coding of the above set of symbols is obvious.

Symbol	Probability	Code
$\boldsymbol{s_1 = 43}$	1/9	11
$\boldsymbol{s_2 = 128}$	3/9	10
$\boldsymbol{s_3 = 171}$	5/9	0

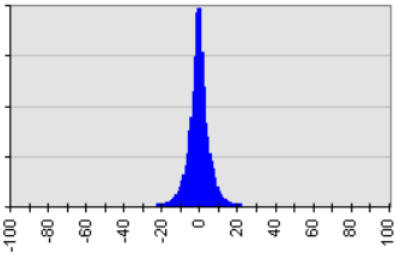
Average number of bits per symbol $l_{avg} = \frac{1}{9} \cdot 2 + \frac{3}{9} \cdot 2 + \frac{5}{9} \cdot 1 = 1.444$ bits/symbol.

(iii) Entropy $H(s) = n = 1.35164$ bits/symbol.
Redundancy $l_{avg} - n = 1.444 - 1.35164 = 0.09236$ or $\frac{l_{avg} - n}{n}\% = 6\%$ of entropy. Huffman code exhibits a low redundancy for the specific alphabet and therefore is efficient enough.

2. (i) A possible histogram of a real-life image is shown below.



(ii) Due to the fact that natural images contain large areas of slowly varying intensity we may assume that image *g* contains a very large number of small values and furthermore it has both positive and negative values which have equal probabilities. Therefore, a possible histogram is shown below.



(iii) Huffman coding $g(x, y)$ instead of $f(x, y)$ is more meaningful since the pdf of $g(x, y)$ is less uniform.

3. (i) The probability of appearance of each intensity is shown in Table 1 below:

Letter	Probability	Codeword
White s_1	0.95	0
Black s_2	0.02	11
Grey s_3	0.03	10

Table 1

The entropy of the source is $H(S) = \sum_{i=1}^{3} p_i I(s_i) = -\sum_{i=1}^{3} p_i \log_2 p_i = 0.335$ bits/symbol.

(ii) The Huffman code is shown in Table 1 above.

(iii) Since we have 3 symbols, a fixed-length code would require $l_{\text{fixed}} = 2$ bits per symbol. With the use of the above Huffman code the average number of bits per symbol is $l_{\text{avg}} = \sum_{i=1}^{3} p_i l_i = 1.05$ bits/symbol.

(ii) The ratio of image size (in bits) between using the fixed length coding and Huffman coding is $\frac{l_{\text{fixed}}}{l_{\text{avg}}} = \frac{2}{1.05} = 1.9$. The coding redundancy is $l_{\text{avg}} - H(s) = 0.715$ bits per symbol.

(iii) The extended by two Huffman code is shown in Table 2 below.

Letter	Probability	Codeword
$s_1 s_1$	0.9025	0
$s_1 s_2$	0.019	111
$s_1 s_3$	0.0285	100
$s_2 s_1$	0.019	1101
$s_2 s_2$	0.0004	110011
$s_2 s_3$	0.0006	110001
$s_3 s_1$	0.0285	101
$s_3 s_2$	0.0006	110010
$s_3 s_3$	0.0009	110000

Table 2

(iv) With the use of the above Huffman code the average number of bits per symbol is $l_{\text{avg}} = \sum_{i=1}^{9} p_i l_i = 0.611$ bits/symbol. The ratio of image size (in bits) between using the fixed length coding and Huffman coding is $\frac{l_{\text{fixed}}}{l_{\text{avg}}} = \frac{2}{0.611} = 3.27$. The coding redundancy is $l_{\text{avg}} - H(s) = 0.276$ bits per symbol.

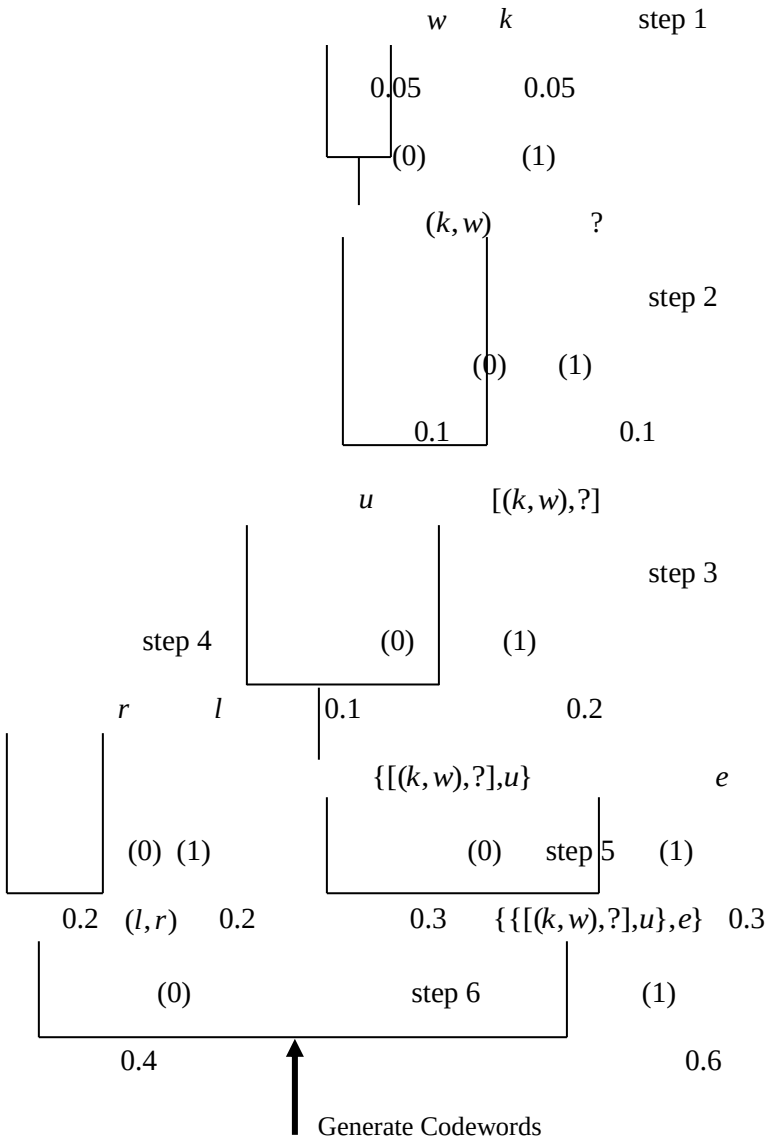
(v) Obviously, the extended Huffman code is more efficient.

4. (i) The discrete memoryless source (DMS) has the property that its output at a certain time does not depend on its output at any earlier time.

(ii) The minimum number of bits per symbol we can achieve is the entropy of the source. In order to achieve this, the probabilities of the symbols must be negative powers of 2.

(iii) This happens when the probabilities of the symbols are equal.

5. (i) While merging branches 1 is preferred to 0 since it has zero probability of being wrongly transmitted and therefore, we assign 1s to branches that correspond to symbols with higher probabilities.



	Step 1	**Step 2**	**Step 3**	**Step 4**	**Step 5**	**Step 6**
k 0.05	e 0.3	e 0.3	e 0.3	e 0.3	(l,r) 0.4	$\{\{[(k,w),?],u\},e\}$ 0.6
l 0.2	l 0.2	l 0.2	l 0.2	$\{[(k,w),?],u\}$ 0.3	e 0.3	(l,r) 0.4
u 0.1	r 0.2	r 0.2	r 0.2	l 0.2	$\{[(k,w),?],u\}$ 0.3	
w 0.05	u 0.1	u 0.1	$[(k,w),?]$ 0.2	r 0.2		
e 0.3	? 0.1	? 0.1	u 0.1			
r 0.2	k 0.05	(k,w) 0.1				
? 0.1	w 0.05					

Symbol	Probability	Codeword
k	0.05	10101
l	0.2	01
u	0.1	100
w	0.05	10100
e	0.3	11
r	0.2	00
?	0.1	1011

(ii) In this example, the average codeword length is 2.6 bits per symbol. In general, the average codeword length is defined as

$$l_{avg} = \sum l_i p_i$$

where l_i is the codeword length (in bits) for the codeword corresponding to symbol s_i. The average codeword length is a measure of the compression ratio. Since our alphabet has seven symbols, a fixed-length coder would require at least three bits per codeword. In this example, we have reduced the representation from three bits per symbol to 2.6 bits per symbol; thus the corresponding compression ratio can be stated as 3/2.6=1.15.

(ii) The probability of 100 to be transmitted correctly is 0.95*0.95=0.9025. Therefore, the probability of 1.

6. (i) 1. To save disk space. 2. To decrease transmission time when transferring files over networks. 3. To make some programs work faster (e.g. by decreasing disk access time).

(ii) It must deviate from uniform.

(iii) After differential coding using the given formula, the resulting image has only 3 values 1, -1, 0 and 0 dominates, therefore it is must more efficient to use differential coding.

7. (i)

$p(110) = 5/25 = 0.2$
$p(120) = 7/25 = 0.28$
$p(140) = 5/25 = 0.2$
$p(160) = 4/25 = 0.16$
$p(170) = 3/25 = 0.12$
$p(180) = 1/25 = 0.04$

(ii) Derive the Huffman code.

Symbol	Prob	Code	Prob	Code	Prob	Code	Prob	Code	Prob	Code
120	0.28	**01**	0.28	01	0.32	00	0.40	1	0.60	0
110	0.2	**10**	0.2	10	0.28	01	**0.32**	00	0.40	1
140	0.2	**11**	0.2	11	**0.2**	10	**0.28**	01		
160	0.16	**000**	**0.16**	000	**0.2**	11				
170	**0.12**	**0010**	**0.16**	001						
180	**0.04**	**0011**								

(iii) Calculate the average length of the fixed code and that of the derived Huffman code.
Fixed length: 3 bits/symbol
Huffman: Lavg=2.48 bits/symbol

(iv) Compression ratio 3/2.48
Redundancy=3-2.48

8. (i) The four reconstruction levels are:

$r_1 = 1/8$ with probability $1/8$

$r_2 = 3/8$ with probability $1/8$

$r_3 = 5/8$ with probability $1/4$

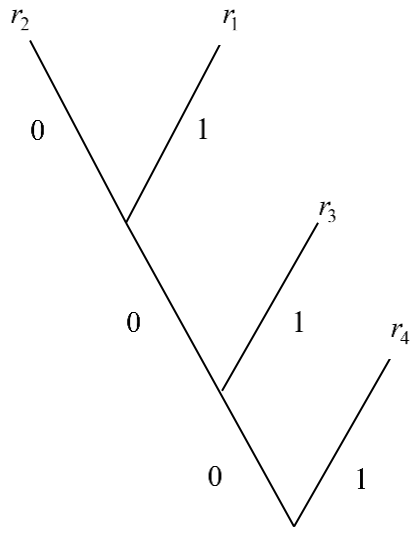
$r_4 = 7/8$ with probability $1/2$

(ii) Uniform quantization does not exploit the pdf of the alphabet to be quantized.

(iii) The Huffman code is found below.

Step 1		Step 2		Step 3	
r_4	1/2	r_4	1/2	r_4	1/2
r_3	1/4	r_3	1/4	$\{r_3,\{r_2,r_1\}\}$	1/2
r_2	1/8	$\{r_2,r_1\}$	1/4		
r_1	1/8				

Symbol	***Codeword***
r_1	001
r_2	000
r_3	01
r_4	1



Average number of bits to represent f

$l_{avg} = \frac{1}{2} + 2\frac{1}{4} + 6\frac{1}{8} = \frac{7}{4}$ bits/symbol

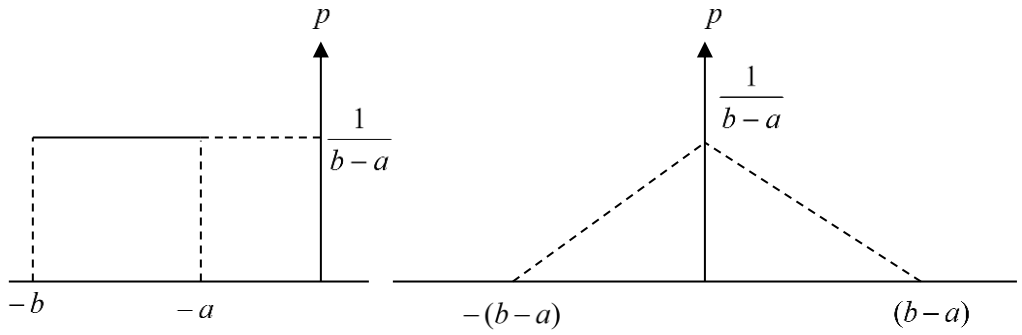
(iv) Entropy $H(s) = -\sum_{i=1}^{3} p_i \log_2(p_i) = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{8} + \frac{1}{8}\log_2\frac{1}{8})$

$= \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = \frac{4}{8} + \frac{4}{8} + \frac{3}{8} + \frac{3}{8} = \frac{14}{8} = \frac{7}{4}$ bits/symbol

Redundancy $= 0$ bits/symbol

Coding efficiency $H(s)/l_{avg} = 100\%$

9.



The pdf of $-f(x-1, y-1)$ is shown on the left and the pdf of $g(x, y)$ is shown on the right (convolution of the 2 pdf's). The pdf of $g(x, y)$ is more skewed and therefore more appropriate for symbol encoding.

10. (i) The integral of the pdf between 0 and 1 must be 1 and according to this restriction we find $c = 5.64$.

(ii) The three reconstruction levels are 1/6, 3/6, 5/6.

(iii) Using Huffman code we get:

Symbol	Probability	Code
$s_1 = 1/6$	0.02	11
$s_2 = 3/6$	0.03	10
$s_3 = 5/6$	0.95	0

Average number of bits per symbol $l_{avg} = 1.05$ bits/symbol.

(iv) Entropy $H(s) = n = 0.335$ bits/symbol.

Redundancy $l_{avg} - n = 1.05 - 0.335 = 0.715$ or $\frac{l_{avg} - n}{n}\% = 213\%$ of entropy! Huffman code exhibits a very high redundancy for the specific alphabet and therefore is not efficient enough.

(v) Suppose we merge the symbols in groups of two symbols. In the next table the extended alphabet and corresponding probabilities and Huffman codewords are shown.

Symbol	Probability	Code
$s_1 s_1$	0.0004	110011
$s_1 s_2$	0.0006	110001
$s_1 s_3$	0.019	1101
$s_2 s_1$	0.0006	110010
$s_2 s_2$	0.0009	110000
$s_2 s_3$	0.0285	101
$s_3 s_1$	0.019	111
$s_3 s_2$	0.0285	100
$s_3 s_3$	0.9025	0

Table: The extended alphabet and corresponding Huffman code

For the new extended alphabet we have
$l_{avg} = 1.222$ bits/**new symbol** or $l_{avg} = 0.611$ bits/original symbol.

Redundancy $\frac{l_{avg} - n}{n}\% = 72\%$ of entropy. Huffman code exhibits a very high redundancy for the specific alphabet and therefore is not efficient enough.