University of London International Programmes
Computing coursework assignments 2015–16

CO3325 Data compression
Coursework assignment 1

This coursework assignment is designed to help you enrich your learning experience and to encourage self-study and creativity. You may write down if necessary additional assumptions in your coursework answers that you employed to simplify your implementation tasks or to help you understand issues. You should, however, attempt the exercises at the end of each chapter in the textbook or subject guide before doing the coursework, otherwise you may find the tasks in the coursework assignment difficult and the experience less rewarding. You should read the coursework assignments or questions carefully and pay particular attention to the Submission Requirements on page 8.
Note: programme source code may be checked using plagiarism detection programs.

Develop a program in Java to implement the following Arithmetic algorithms for compression and decompression.

Let $S = (s_1, s_2, \cdots, s_n)$ be the alphabet of a source with an associated probability distribution of occurrence $P = (p_1, p_2, \cdots, p_n)$. The subintervals for each iteration can be derived for every symbol according to these probabilities. For example, after the first iteration, the initial iteration [0, 1) can be divided into n intervals as below:

$$[0,\ p_1)$$

$$[p_1,\ \underline{p_1} + p_2)$$

$$[p_1 + p_2,\ \underline{p_1 + p_2} + p_3)$$

$$\vdots$$

$$[p_1 + p_2 + \cdots + p_{n-1},\ \underline{p_1 + p_2 + \cdots + p_{n-1}} + p_n)$$

where $p_1 + p_2 + \cdots + p_n = 1$ for the n symbols in the alphabet $s_1, s_2, \cdots, s_n$ respectively. If the first symbol read is the ith symbol s_i in the alphabet, then the left boundary of the subinterval `low` is the cumulative probability $P_i = p_1 + p_2 + \cdots + p_{i-1}$ and the right boundary `high` is $\texttt{low} + p_i$.

The length of the interval $\texttt{high} - \texttt{low}$ can then be used to compute one of the intervals in the next iteration:

$$[\texttt{low},\ \texttt{low} + (\texttt{high} - \texttt{low})P_1)$$

$$[\texttt{low+(high-low)}P_1,\ \texttt{low} + (\texttt{high} - \texttt{low})P_2)$$

$$\vdots$$

$$[\texttt{low} + (\texttt{high} - \texttt{low})P_{n-1},\ \texttt{low} + (\texttt{high} - \texttt{low})P_n)$$

where the cumulative probability on the whole set is $P_n = p_1 + p_2 + \cdots + p_n = 1$.

This looks complicated but there is no need to compute all the subintervals except the one which depends on the independent probability of the symbol read at that iteration. As we can see from the algorithm later, the variables `low`, `high`, and the cumulative probability P_i can be easily updated.

The arithmetic algorithm for compression and decompression is given below: we use two variables `low` and `high` to define the arithmetic interval [`low, high`), where the interval can be easily changed by updating the values of `low` or `high`.

Let *low, high*, and *range* be a real.

Algorithm 1 Arithmetic Compression

```
1: $low \leftarrow 0.0$
2: $high \leftarrow 1.0$
3: while there are still input symbols do
4:     get an input symbol $s$
5:     $range \leftarrow high - low$
6:     $high \leftarrow low + range \times high\_range(s)$
7:     $low \leftarrow low + range \times low\_range(s)$
8: end while
9: output $low$
```

end

Algorithm 2 Arithmetic Decompression

```
1: get encoded number
2: repeat
3:     find symbol whose range covers the encoded number
4:     output the symbol
5:     $range \leftarrow symbol\_high\_value - symbol\_low\_value$
6:     subtract $symbol\_low\_value$ from $encoded\text{-}number$
7:     divide $encoded\_number$ by range
8: until no more symbols
```

end

There are no specific requirements for the user interface apart from a basic implementation as the focus of the coursework is on demonstrating your knowledge and understanding of the algorithms. The following example shows a simple user interface of such a basic Java program running on a binary alphabet (text after ">>" input by the user):

```
2910325 Data Compression Coursework 1
by xxxx (your FAMILYNAME-firstname_SRN)
ARITHMETIC COMPRESSION
**********************
1. Encoding
2. Decoding
0. Exit
**********************
Please input a single digit (0-2) :
>> 1
Enter a String From A BINARY Alphabet to Encrypt
>> yxyxy
```

```
Enter The Probability of : y
Please Do Not Enter As Follows, e.g. 1/2 , 2/3
Enter In The Following Format, e.g. 0.25, 0.776
>> 0.7
S1 = x
S2 = y
Next Symbol: y
L = 0.30000000000000004 D = 0.7
Next Symbol: x
L = 0.30000000000000004 D = 0.21000000000000002
Next Symbol: y
L = 0.36300000000000004 D = 0.147
Next Symbol: x
L = 0.36300000000000004 D = 0.04410000000000001
Next Symbol: y
L = 0.37623000000000006 D = 0.03087
The Output is : 0.39769083

ARITHMETIC COMPRESSION
**********************
1. Encoding
2. Decoding
0. Exit
**********************
Please input a single digit (0-2) :
>> 2
Please Enter The Two Characters One By One
>> b
>> a
Please Enter The Number Of Encoded Symbols
>> 5
Please Enter The Probability Of : b
>> 0.7
Please Enter The Output Value Of The Encoded Symbols
>> 0.39769083
L = 0.30000000000000004, D = 0.7
L = 0.30000000000000004, D = 0.21000000000000002
L = 0.36300000000000004, D = 0.147
L = 0.36300000000000004, D = 0.04410000000000001
L = 0.37623000000000006, D = 0.03087
Decoded Symbols Are : babab

ARITHMETIC COMPRESSION
**********************
1. Encoding
2. Decoding
0. Exit
**********************
```

```
Please input a single digit (0-2) :
>> 0
Thank you and good bye!
```

Alternatively, you may design a concise user interface in order to report iterative execution states of the algorithms in a table, tracing the values of $L, \delta, \delta \times p_x, \delta \times p_y$, and the `output-interval` as in the following example for compression (let $p_x = 0.2$ and $p_y = 0.8$):

Stage	Next-char	L	δ	$\delta \times p_x$	$\delta \times p_y$	Output-interval
0		0	1			
1	b	0.2	0.8	0.2	0.8	
2	a	0.2	0.16	0.16		
3	b	0.232	0.128	0.032	0.128	
4	a	0.232	0.0256	0.0256		
5	b	0.23712	0.02048	0.00512	0.02048	[0.23712, 0.2576)

A basic Java program would be able to show the *fast-approaching-to-zero* implementation problem of the Arithmetic encoding algorithm by the input of a long string such as `ababaaabbbabbbbbb`. An advanced Java program would, however, be able to demonstrate that the problem may be solved to a certain degree by using the following user interface like the one below that offers an extended menu of execution:

```
ARITHMETIC COMPRESSION
**********************
1. Encoding Basic
2. Decoding Basic
3. Encoding Enhanced
4. Decoding Enhanced
0. Exit
**********************
```

The source code of a basic program is worth up to [40%] credits but the source code of an advanced program is worth up to [60%].

[END OF COURSEWORK ASSIGNMENT 1]

University of London International Programmes
Computing coursework assignments 2015–16

CO3325 Data compression
Coursework assignment 2

This coursework assignment is designed to help you enrich your learning experience and to encourage self-study and creativity. You may write down if necessary additional assumptions in your coursework answers that you employed to simplify your implementation tasks or to help you understand issues. You should, however, attempt the exercises at the end of each chapter in the textbook or subject guide before doing the coursework, otherwise you may find the tasks in the coursework assignment difficult and the experience less rewarding. You should read the coursework assignments or questions carefully and pay particular attention to the Submission Requirements on page 8.
Note: programme source code may be checked using plagiarism detection programs.

Develop a prototype Java program to demonstrate how the Hierarchical (Pyramid) Coding algorithms work.

Pyramid Coding is a progressive image compression technique for the efficient transmission of large images to remote users. Typically users are selecting from a large number of images which are being sent to them. To speed up the selection process, a viewer would prefer to receive highlights of each image promptly rather than each detailed image slowly. Hence, instead of sending the pixels of an image line by line, the Pyramid Coding sends a sequence of 'representatives' of the image pixels as the highlights.

One way to generate such a highlight sequence is to divide the original image into four blocks of subimages and let the *top-left* pixel of each subimage be a representative. Hence the first element of the sequence would be a matrix of 4 representatives after the first-time division. Next, each subimage can be further divided into four subsubimages, so the second element of the sequence would be a matrix of $4 \times 4 = 4^2$ representatives after the second-time divisions. Next, each subsubimage can be further divided into four subsubsubimages, so the third element of the sequence would be a matrix of $4 \times 4^2 = 4^3$ representatives after the third-time divisions. This process may continue until the image cannot be divided further.

An image can be modelled as a matrix of decimal integers (for ease of discussion) and each integer entry ($\in [0, 2^{24}]$) of the matrix represents a 3-byte colour code (Red, Green, Blue). The elements of the sequence, i.e. the matrices of representatives, if placed one after another aligned vertically in a diagram, would appear as a pyramid of representatives. Each element of the sequence (representative matrix) would therefore be referred to as a *layer* of the pyramid. As such, the jth division would generate the Layer-j of the pyramid consisting of 4^j representatives of the image.

For example, consider the 4×4 image as follows:

$$M_{4\times4} = \begin{pmatrix} 0 & 0 & 2 & 7 \\ 0 & 1 & 2 & 5 \\ 0 & 1 & 2 & 7 \\ 0 & 4 & 2 & 7 \end{pmatrix}$$

The image $M_{4\times4}$ can be divided into four blocks of 2×2 subimage matrices:

$$\begin{array}{cccc} 0 & 0 & 2 & 7 \\ 0 & 1 & 2 & 5 \\ 0 & 1 & 2 & 7 \\ 0 & 4 & 2 & 7 \end{array} \rightarrow \begin{array}{cc|cc} \mathit{0} & 0 & \mathit{2} & 7 \\ 0 & 1 & 2 & 5 \\ \hline \mathit{0} & 1 & \mathit{2} & 7 \\ 0 & 4 & 2 & 7 \end{array}$$

Using the top-left element as the representative for each block, the image can be approximated via $M_{4\times4} = \begin{pmatrix} \mathit{0} & 2 \\ 0 & 2 \end{pmatrix}$ or the four representatives (from left to right, and top to bottom): (0, 2, 0, 2)

Upon receipt of B_1, the representative matrix, the decoder may convert

$$\begin{matrix} \mathit{0} & 2 \\ 0 & 2 \end{matrix} \Rightarrow \begin{array}{cc|cc} \mathit{0} & 0 & \mathit{2} & 2 \\ 0 & 0 & 2 & 2 \\ \hline \mathit{0} & 0 & \mathit{2} & 2 \\ 0 & 0 & 2 & 2 \end{array} \rightarrow \begin{matrix} 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{matrix}$$

This is a so-called '2-layer pyramid structure' with the block size 2×2 on layer-1 (approximation) and the block size 4×4 (original) on layer-2.

In this example, the input of the Pyramid encoding is the 4×4 matrix $\begin{matrix} 0 & 0 & 2 & 7 \\ 0 & 1 & 2 & 5 \\ 0 & 1 & 2 & 7 \\ 0 & 4 & 2 & 7 \end{matrix}$.

The output is the sequence of the two matrices: $\begin{matrix} \mathit{0} & 2 \\ 0 & 2 \end{matrix}$ and $\begin{matrix} 0 & 0 & 2 & 7 \\ 0 & 1 & 2 & 5 \\ 0 & 1 & 2 & 7 \\ 0 & 4 & 2 & 7 \end{matrix}$.

The input of the Pyramid decoding is the sequence of the two matrices $\begin{matrix} \mathit{0} & 2 \\ 0 & 2 \end{matrix}$ and $\begin{matrix} 0 & 0 & 2 & 7 \\ 0 & 1 & 2 & 5 \\ 0 & 1 & 2 & 7 \\ 0 & 4 & 2 & 7 \end{matrix}$.

The output of the Pyramid decoding is the sequence of the two matrices $\begin{matrix} 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{matrix}$ and $\begin{matrix} 0 & 0 & 2 & 7 \\ 0 & 1 & 2 & 5 \\ 0 & 1 & 2 & 7 \\ 0 & 4 & 2 & 7 \end{matrix}$.

Similarly, a '3-layer pyramid' can be derived for

$$M_{8\times8} = \begin{pmatrix} 0 & 0 & 0 & 0 & 2 & 4 & 7 & 7 \\ 0 & 0 & 0 & 0 & 2 & 5 & 6 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 5 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 6 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 7 & 7 \\ 0 & 0 & 3 & 3 & 2 & 4 & 6 & 7 \\ 0 & 0 & 4 & 2 & 2 & 3 & 7 & 7 \\ 0 & 0 & 5 & 1 & 2 & 2 & 7 & 7 \end{pmatrix}$$

$M_{8\times8}$ can be divided into four 4×4 sub-matrices A–D:

$$\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 2 & 4 & 7 & 7 \\ 0 & 0 & 0 & 0 & 2 & 5 & 6 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 5 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 6 & 7 \\ \hline 0 & 0 & 1 & 1 & 2 & 2 & 7 & 7 \\ 0 & 0 & 3 & 3 & 2 & 4 & 6 & 7 \\ 0 & 0 & 4 & 2 & 2 & 3 & 7 & 7 \\ 0 & 0 & 5 & 1 & 2 & 2 & 7 & 7 \end{array} = \begin{array}{c|c} A & B \\ \hline C & D \end{array}$$

where

$$A = \begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{array} \quad B = \begin{array}{cc|cc} 2 & 4 & 7 & 7 \\ 2 & 5 & 6 & 7 \\ \hline 2 & 2 & 5 & 7 \\ 2 & 2 & 6 & 7 \end{array} \quad C = \begin{array}{cc|cc} 0 & 0 & 1 & 1 \\ 0 & 0 & 3 & 3 \\ \hline 0 & 0 & 4 & 2 \\ 0 & 0 & 5 & 1 \end{array} \quad D = \begin{array}{cc|cc} 2 & 2 & 7 & 7 \\ 2 & 4 & 6 & 7 \\ \hline 2 & 3 & 7 & 7 \\ 2 & 2 & 7 & 7 \end{array}$$

There are (3 pyramid layers) layer-1, layer-2 and layer-3, for blocks of sizes 2×2, 4×4 and 8×8, respectively:

$$(1) \begin{array}{cc} 0 & 2 \\ 0 & 2 \end{array} \quad (2) \begin{array}{cc|cc} 0 & 0 & 2 & 7 \\ 0 & 1 & 2 & 5 \\ \hline 0 & 1 & 2 & 7 \\ 0 & 4 & 2 & 7 \end{array} \quad (3) \begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 2 & 4 & 7 & 7 \\ 0 & 0 & 0 & 0 & 2 & 5 & 6 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 5 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 6 & 7 \\ \hline 0 & 0 & 1 & 1 & 2 & 2 & 7 & 7 \\ 0 & 0 & 3 & 3 & 2 & 4 & 6 & 7 \\ 0 & 0 & 4 & 2 & 2 & 3 & 7 & 7 \\ 0 & 0 & 5 & 1 & 2 & 2 & 7 & 7 \end{array}$$

The decoding process for (1)–(3) is progressive and can be terminated by the decoder as soon as the displayed image (4), (5) or (6) is considered 'good enough'.

$$(4) \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \end{array} \quad (5) \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 2 & 2 & 7 & 7 \\ 0 & 0 & 0 & 0 & 2 & 2 & 7 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 5 & 5 \\ 0 & 0 & 1 & 1 & 2 & 2 & 5 & 5 \\ 0 & 0 & 1 & 1 & 2 & 2 & 7 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 7 & 7 \\ 0 & 0 & 4 & 4 & 2 & 2 & 7 & 7 \\ 0 & 0 & 4 & 4 & 2 & 2 & 7 & 7 \end{array} \quad (6) \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 2 & 4 & 7 & 7 \\ 0 & 0 & 0 & 0 & 2 & 5 & 6 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 5 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 6 & 7 \\ 0 & 0 & 1 & 1 & 2 & 2 & 7 & 7 \\ 0 & 0 & 3 & 3 & 2 & 4 & 6 & 7 \\ 0 & 0 & 4 & 2 & 2 & 3 & 7 & 7 \\ 0 & 0 & 5 & 1 & 2 & 2 & 7 & 7 \end{array}$$

In the above example, the input of the Pyramid encoding is matrix $M_{8\times 8}$. The output of the Pyramid encoding is a sequence of three matrices (1), (2) and (3).

The input of the Pyramid decoding is the sequence of the three matrices (1), (2) and (3). The output of the Pyramid decoding is a sequence of three matrices (4), (5) and (6).

A good implementation program would be able to deal with an input matrix of any suitable size n, but a limited-input-size implementation $n \leq 4$ is acceptable provided that clear instructions are given to the user of your prototype program. Advanced students are encouraged (but not required) to explore and discuss compression issues in the implementation if time permits. For example, a representative in the Pyramid coding does not have to be one of the pixels of the input image. Instead, the JPEG rules on page 86 of the CO3325 subject guide (2004) may be applied. Also, the representatives do not have to be sent repeatedly.

The source code of a basic demonstrate program is worth up to [40%] credits and the source code of an advanced program is worth up to [60%].

[END OF COURSEWORK ASSIGNMENT 2]

Submission requirements

These requirements apply to both coursework assignments. The available marks are given in square brackets.

1. Naming conventions for any `.pdf` or `.zip` file submissions
   When naming your files, please ensure that you include your full name, student number, course code and assignment number, e.g. `FamilyName_SRN_COxxxxcw#.pdf`
   (e.g. `Zuckerberg_920000000_CO3325cw2.pdf`), where
   - `FamilyName` is your family name (also known as last name or surname) as it appears in your student record (check your student portal)
   - `SRN` is your Student Reference Number, for example 920000000
   - `COXXXX` is the course number, for example CO3325, and cw# is either cw1 (coursework 1) or cw2 (coursework 2).
2. Your coursework submission must include a report Document [40%] and the program Code [60%].
   The Document (preferable in .pdf format) should include the following sections:
   (a) Algorithms (in flow-chart)
   (b) Design (in block diagram or class-diagram in UML)
   (c) Demonstration (in 5 best screen-shots)
   (d) Discussion (including answers to any questions/problems in the Coursework assignment, your experience in attempt of the coursework, and full bibliography)

   The program code should include the
   (a) Java source codes .java
   (b) executable version .class.
3. Execution of your programs:
   [Penalty] A ZERO mark may be awarded if
   - your program(s) cannot be run from the coursework directory by a simple command '*java menu*' (this means that you should name your main class 'menu', or adopt the `menu.java` that can be found in the Appendix on page 9);
   - your source code(s) does not compile and you give no information on your program execution environment;
   - your program(s) does not do what you claim it should do;
   - your program(s) crashes within the first *three* interactive execution steps;
   - your program(s) works for the first time of execution only;
   - there is no comment in your source code.
4. You should monitor and report the time you have spent for each part of the coursework answers, and leave a note to the examiner if you need to raise any issue at the beginning of your coursework answers as follows:

Total Number of Hours Spent	
Hours Spent for Algorithm Design	
Hours Spent for Programming	
Hours Spent for Writing Report	
Hours Spent for Testing	
Note for the examiner (if any):	

5. Show *all* your work. Any use of others' work should be declared at the point of use and referred to in the *Bibliography* section at the end of your coursework answers.

Appendix

This is an example. Please modify accordingly to suit your own purposes.

```
import java.lang.*;
import java.io.*;
// Modify the display content to suit your purposes...
class menu {
private static final String TITLE =
"\nCO3325 Data Compression coursework\n"+
"   by FAMILYNAME-firstname_SRN\n\n"+
"\t*******************\n"+
"\t1. Declaration: Sorry but part of the program was copied
from the Internet! \n" +
"\t2. Question 2 \n"+
"\t3. Question 3 \n"+
"\t4. no attempt \n"+
"\t0. Exit \n"+
"\t*******************\n"+
"Please input a single digit (0-4):\n";
menu() {
int selected=-1;
while (selected!=0) {
System.out.println(TITLE);
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
// selected = Integer.parseInt(in.readLine());
try {
                       selected = Integer.parseInt(in.readLine());

                           switch(selected) {
                                case 1:  q1();
                                  break;
                                case 2:  q2();
                                  break;
                                case 3:  q3();
                                  break;
                                case 4:  q4();
                                  break;}      }
   catch(Exception ex) {}  } // end while
                System.out.println("Bye!");
}
// Modify the types of the methods to suit your purposes...
private void q1() {
System.out.println("in q1");
}
private void q2() {
System.out.println("in q2");
}
private int q3() {
System.out.println("in q3");
return 1;
}
private boolean q4() {
System.out.println("in q4");
return true;
}
   public static void main(String[] args) {
new menu();
   }
}
```

[END OF SUBMISSION REQUIREMENTS]