**University of London**
**Computing and Information Systems/Creative Computing**
**CO3325 Data compression**
**Coursework assignments 1 and 2 2021–2022**

## Important Information

Both coursework assignments are 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 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 assignments difficult and the experience less rewarding. You should read the coursework assignments carefully and pay particular attention to the submission requirements.

## Important reminder

It is important that your submitted coursework assignment is your own individual work and, for the most part, written in your own words. You must provide appropriate in-text citation for both paraphrase and quotation, with a detailed reference section at the end of your assignment (this should not be included in the word count). Copying, plagiarism, unaccredited and/or wholesale reproduction of material from books or from any online source is unacceptable, and will be penalised (see: How to avoid plagiarism). You may also find it helpful to look at the end of some journal or conference papers to get an idea of how to cite and reference your material appropriately. The Harvard Referencing Guide provides short explanatory introduction and a checklist of examples, showing how to cite and reference material from various sources.

## Submission requirements

These requirements apply to both coursework assignments.

1. **Report and code**
   Your coursework submission worth 100 marks must include a report [40 marks] and the program code [60 marks].

   The report must be presented in a `.pdf` file, and should include the following sections:

   (a) Design (in block diagram or class-diagram in UML)
   (b) Algorithms (in flow-chart)
   (c) Demonstration (in **FIVE** best screen-shots)
   (d) Discussion (including answers to any questions/problems in the coursework assignment, your experience in attempting the coursework, citations and a reference section in Harvard format)

   The code must be presented in a zipped file (e.g. `.zip`) if there are multiple Java source codes and auxiliary files, and should include:

   - Java source codes in `.java` files
   - Any other files, e.g. `inputfile.txt` and `outputfile.txt`.

2. **File names**
   Naming conventions for any `.pdf` (or `.zip`) file submissions must be in the form of `YourName_SRN_CO3325cw#.pdf` (e.g. `MaryBrown_920000000_CO3325cw2.pdf`), where

- `YourName` is your name as it appears on your student record (check your student portal), e.g. `MaryBrown`.
- `SRN` is your Student Reference Number, e.g. `920000000`.
- `CO3325` is the course number, and `cw#` is either `cw1` (coursework 1) or `cw2` (coursework 2).

The main Java class of your program codes must be included in the file named as '`menu.java`' (so the class file `menu.class` can be tested by examiners using the single command line '`java menu`'). You may use any names for other Java codes. The Appendix on page 5 shows an example in which four Java codes `q1()`–`q4()` can be integrated and run selectively after the command line '`java menu`'. You are encouraged to include the `.class` files in your submission (not required but just in case).

**3. Execution of your programs**

Penalty (a mark of zero) may be awarded if:

- your program(s) cannot be run using the command line '`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 5)
- 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 are no comments in your source code.

**4. Timing**

You should monitor and report the time you have spent on each part of the coursework. Please leave a note to the examiners if you need to raise any issue at the **beginning** of your submission 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 examiners (if any): | |

**5. Citation and referencing**

Show *all* your work. Any use of others' work should be declared at the point of use and referred to in the Reference section at the end of your submission.

**[END OF SUBMISSION REQUIREMENTS]**

**Coursework assignment 1**

Develop Java prototype programs/methods for compression algorithms/programs.

1. Develop a Java program method to display the first[1] longest match of a string of characters.

   The method takes two strings, namely *long* and *short*, as the input, where *long* contains no fewer characters than *short* (i.e. the length of string *long* is greater or equal to that of *short*). The method will then return a longest match substring of *short*. Assume the strings are read from left to right. For example:

| *long* | *short* |
|---|---|
| 12345678901234567890123456789012345678901 | 1234567890123456 |
| this␣is␣xx␣an␣example␣and␣another␣example | xxx␣n␣example␣xx |

   "n␣example" and index location "13–21" in *long*, and "5–13" in *short* should be returned.

2. Further to the above task, develop a Java method to display *all* the longest match(es) of a string and the number of these matches (0 for 'no match'). Assume the strings are read from right to left (different from the previous task).

3. Add a function to your Java program so it can replace the matches in *long* with blanks, that is, remove the matches and keep the matched positions as blanks.

**Hints**: You need a proper design for your algorithm/program in diagrams *before* any implementation. There are many decisions to be made and many questions to be answered. For example, what are the main functions of my prototype/program? What is the input/output? What extra information does my prototype offer the user? What data structures should I use to make my prototype program more efficient?

There are often many ways to implement a data structure. For example, you could use a linked list, array, tree, heaps, tries, etc.Therefore, you may need to review your work in the Software engineering, algorithm design and analysis course (see Volume 2 of the subject guide CO2226 Software engineering, algorithm design and analysis).

You are encouraged to explore and highlight (in the discussion section of your report) various implementation details. For example, assume that the plain English text is stored in the source file `inputfile.txt`, and the compressed version is in `outputfile.txt`. You may add more assumptions to ease your programming tasks.

**Note**: if you use others' program code (including parts of others' code), you must identify this code and its source both in your report (section Discussion) and in your code (on the Main Menu).

**[END OF COURSEWORK ASSIGNMENT 1]**

[1] i.e. first-time found

**Coursework assignment 2**

Implement LZ77 compression (and decompression) algorithms.

1. Implement in Java an LZ77 (or LZSS) encoding and decoding algorithm to visualise the execution of the algorithm. Your program should be able to run stepwise as shown in examples provided in the CO3325 subject guide.

   Note:

   (a) Your submission should include an LZ77 algorithm as comments at the beginning of your source code.

   (b) You should keep a sets of dated programming logs to demonstrate how you have progressed from the start of program design to the end of the implementation.

2. Further to the above task, add a function to your Java program so it can also display the compression ratio after the compression.

**Hints**: You need a proper design for your compressor/decompressor in diagrams *before* any implementation. There are many decisions to be made and many questions to be answered. For example, what are the main functions of my prototype/program? What is the input/output? What constitutes a basic compressor or decompressor that returns an encoded or decoded message? How can it also allow the user to run my prototype step-by-step with pauses? What extra information does my prototype offer the user? Can my prototype report to the user any other useful information along with the compression ratio? What data structures can I use to make my prototype program more efficient?

There are often many ways to implement a data structure. For example, you could use a linked list, array, tree, heaps, tries, etc.Therefore, you may need to review your work in the Software engineering, algorithm design and analysis course (see Volume 2 of the subject guide CO2226 Software engineering, algorithm design and analysis).

You are encouraged to explore and highlight (in the discussion section of your report) various implementation details. For example, assume that the plain English text is stored in the source file `inputfile.txt`, and the compressed version is in `outputfile.txt`. You may add more assumptions to ease your programming tasks.

**Note**: if you use others' program code (including parts of others' code), you must identify this code and its source both in your report (section Discussion) and in your code (on the Main Menu).

**[END OF COURSEWORK ASSIGNMENT 2]**

## Appendix

This is an example of `menu.java`. You may modify it as appropriate to suit your own purposes.

The `menu.class` should be run using the single command line '`java menu`'. It displays a "menu" of four choices and allows the user to selectively run the four Java codes `q1(), q2(), q3()` and `q(4)` accordingly.

```
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"+
"\t0. 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 APPENDIX]**