**SCHOOL OF COMPUTER SCIENCE**
**COURSEWORK ASSESSMENT PROFORMA**

**MODULE & LECTURER:** CM1210 Object-Oriented Java Programming; Louise Knight & Polona Štefanič

**DATE SET:** 16th April 2018

**SUBMISSION DATE:** 4th May 2018 at 9:30am

**SUBMISSION ARRANGEMENTS:** Submission of all material (short report and Java code) via Learning Central.

**TITLE:** Practical implementation of algorithms and data structures

This coursework is worth 50% of the total marks available for this module. The penalty for late or non-submission is an award of zero marks. You are reminded of the need to comply with Cardiff University's Student Guide to Academic Integrity. Your work should be submitted using the official Coursework Submission Cover sheet.

**INSTRUCTIONS**

A) Write a method *parseData*(*data*) that parses the file data.txt (available on Learning Central) and returns in an *ArrayList*, at most 1000 words that are longer than 3 characters. We recommend that the data is stored in an *ArrayList* within the Java program. (This part should be achievable now.)

B) Implement the two algorithms bubble sort and merge sort in order to sort the words in alphabetical order (the pseudocode for these methods is available in the lecture notes). For each of the algorithms write a separate Java method e.g. *bubbleSort*(*data*) and *mergeSort*(*data*). For each of the sorting algorithms measure

- time that is needed to sort 100 of the words, 200 of the words, 300 of the words, etc. up to all 1000 words by each of the two algorithms
- count the moves and swaps that occur while sorting elements

(This part should be achievable now. Before attempting this exercise you should work through the Algorithms lab exercises, available on Learning Central. The techniques used there will help you to work out how to approach this part of the coursework, in particular there are examples of how to time algorithms and count the moves and swaps.)

C) You should create two methods for a data structure of *LinkedList*. Your data structure should have the class name *MyLinkedList*. The two methods that should be implemented are:

• Adding element at specific location in the list:

```
public void addAtPosition(int position, String item) {…}
```

• Deleting an element from the list at specific position where the method should return which Node is deleted:

```
public Node deleteAtPosition(int position) {…}
```

In both methods handle error properly (e.g. what happens if position does not exists).

There will be a skeleton code of MyLinkedkList available on Learning central that contains: Node class, MyLinkedList class with calculateSize(), addFirst(), addLast(), traverse(), findByPosition() methods and signatures for two methods addAtPosition(int position, String item), deleteAtPosition(int position) that you should implement.

You can reuse any implemented methods in the skeleton code.
You ARE NOT allowed to change any parts of the methods.

(This part should be achievable after the lectures on April 19.)

All code must be written by you, although you can use the lecture notes (and lab exercises), textbooks, and the Oracle Java website for guidance.

**SUBMISSION INSTRUCTIONS**

You should upload the following to Learning Central. Note that any deviation from this will result in a mark of zero for the question(s) affected.

Your solution must include:

• A filled-in copy of the official Coursework Submission Cover sheet
• A typeset PDF report (approximately 1000 words/roughly 3-4 pages of text; diagrams, screenshots, charts etc. are encouraged) detailing the following:
  - An introduction to the report as a whole, specifying what you shall be presenting in the report
  - A basic algorithmic description of how each sorting algorithm works
  - An overview of your program design and implementation (include code snippets for each sorting algorithm and each of the points in part C above, and briefly describe what each snippet does)
  - Results of the timing and move/swap count experiments for each sorting algorithm (as tables, graphs, etc. as appropriate)
  - A brief user guide describing how to run the sorting algorithms
  - An explanation of how each of the above methods work
  - How to run your code (provide this in a “README” file)
• All Java source files, and related files (e.g. dependent third-party libraries and graphics), submitted as a single zip file collection.

Ensure that your student number is included as a comment at the top of each Java file that makes up your submission.

| Description | | Type | Name |
|---|---|---|---|
| Cover sheet | **Compulsory** | PDF (.pdf) file | [student number].pdf |
| ***ONE*** PDF file (and no more than one) which contains the report points listed above ("A typeset PDF report…") | **Compulsory** | PDF (.pdf) file | CM1210_[student number].pdf |
| ***ONE*** ZIP archive (and no more than one) of all the source code written | **Compulsory** | ZIP (.zip) archive | CM1210Source_[student number].zip |

## CRITERIA FOR ASSESSMENT

Credit will be awarded against the following criteria.

- *parseData*(*data*) method - 5 marks (and 3 marks for description of how code works)
- Bubble sort implementation - 6 marks (and 3 marks for description of how code works)
- Merge sort implementation - 6 marks (and 3 marks for description of how code works)
- *MyLinkedList* – creation of the instance - 4 marks
- *addAtPosition()* implementation - 7 marks
- *deleteAtPosition()* implementation - 7 marks
- Report introduction - 3 marks
- Description of bubble sort algorithm - 5 marks
- Description of merge sort algorithm - 5 marks
- Description of My*LinkedList* methods implementation (how you implemented)
  - *public void addAtPositiont*(int position, String item) – 6
  - *public Node deleteAtPosition*(int position) – 6
- Results of timing experiments (for algorithms only) - 8 marks
- Results of move/swap experiments - 10 marks
- User guide for how to use each program - 3 marks

(The code provided may be run to check it works properly, and will be manually inspected.)

- Readability / Interpretation of the results shown - 5
- Report conclusion / summary - 5 marks

(Total: 100 marks)

Feedback on your performance will address each of these criteria.

**FURTHER DETAILS**

Feedback on your coursework will address the above criteria and will be returned in approximately: 3 weeks

Individual feedback will be given via Grade Centre.