CM10228 Coursework 2
Problem Solving in C and Java

March 25, 2019

1 Introduction

The coursework of this unit consists of two large courseworks. This document provides the specification for the second larger coursework. Please read carefully and take note of the source code that has been provided, and for some questions example input files, as we will use these files to test/ mark your code.

2 Problems

The coursework contains 8 questions for you to solve programmatically. Each question should be answered in a separate file named (depending on the question) CW2Q1.java (or CW2Q2.c) and be placed in the root of the folder you submit.

The learning outcomes of this coursework include understanding the underlying implementations of built in methods and functions. This allows you to gain a better understanding of "how" these methods/-functions might have been implemented and the challenges the programmer might face when developing an optimal solution in terms of speed of execution or memory allocation etc.

Furthermore, by understanding how these methods/functions might have be implemented it allows you to make a comparison between the styles of the two languages (i.e. C and Java) and the advantages and limitations they provide.

Therefore, for all of the questions below effort should be made to understand the basic implementations of the methods/ functions by developing your own implementations of inbuilt and imported methods/functions. For example, in Q1 you might use the inbuilt method `.compareTo()` in the String class, however, you are encouraged to

(re)implement your own version of this method.

Your (re)implementation of these inbuilt methods/functions is reflected in the marks for *Correctness* i.e. 40% of each question. By using only inbuilt methods/functions you can achieve half of the correctness marks. Implementing one or more of your own methods/-functions to substitute inbuilt or imported library methods/functions will count towards the remaining half of the correctness marks i.e. a max of 20% for each question.

2.1 Problem Questions

Q1. Implement a bubble sort algorithm for a list of names in Java.

Q2. Implement a quick sort algorithm for a list of names in C.

Q3. Implement your own Hash Table in C for storing and searching for names i.e. strings or char arrays. Your Hash Table implementation should have the following interface written below in pseduocode. ***Note: how you define the structures, functions, hashing algorithm and the limit (if any) of the number of items that can be stored in the Hash Table etc. is up to you.***

```
void add(name);

void remove(name);

boolean search(name);
```

Q4. Implement your own XOR Linked List in Java capable of storing names. Your implementation should have the following method, written below in pseduocode,. ***Note: how you define the methods and classes needed to implement you XOR Linked List etc. is up to you.***

```
public void insertAfter(String after, String newObj)

public void insertBefore(String before, String newObj)

public String removeAfter(String after)

public String removeBefore(String before)
```

NOTE: For questions 1 to 4 you should test your algorithms with the list of 5000+ names provided.

Q5. Implement an algorithm in C which given a block of text as input, redacts all words from a given set of "redactable" words. For example, given the block of text,

The quick brown fox jumps over the lazy dog

and the redactable set of words

the, jumps, lazy

the output text should be

**** quick brown fox ***** over *** **** dog*

Note that the number of stars in the redacted text is the same as the number of letters in the word that has been redacted, and that capitalization is ignored. You should **not** use any of the string libraries to answer this question. You should also test your program using the example files provided.

Q6. Implement in Java a similar algorithm to that in Q5 i.e. given a block of text your algorithm should be able to redact words from a given set. However, in this implementation of the algorithm all redactable words will be proper nouns (i.e. a name used for an individual person, place, or organisation, spelled with an initial capital letter) and your algorithm should take into account that the list of redactable words might not be complete. For example, given the block of text,

It was in July, 1805, and the speaker was the well-known Anna Pavlovna Scherer, maid of honor and favorite of the Empress Marya Fedorovna. With these words she greeted Prince Vasili Kuragin, a man of high rank and importance, who was the first to arrive at her reception. Anna Pavlovna had had a cough for some days. She was, as she said, suffering from la grippe; grippe being then a new word in St. Petersburg, used only by the elite.

and the redactable set of words

Anna Pavlovna Scherer, St. Petersburg, Marya Fedorovna

the output text should be

*It was in ****, 1805, and the speaker was the well-known **** ******** *******, maid of honor and favorite of the ******* ***** *********. With these words she greeted ****** ****** *******, a man of high rank and importance, who was the first to arrive at her reception. **** ******** had had a cough for some days. She was, as she said, suffering from la grippe; grippe being then a new word in *** *********, used only by the elite.*

You should test your program using the example files provided.

Q7. Implement a Columnar Transposition Cipher in C to encrypt a message of any length. A Columnar Transposition Cipher is transposition cipher that follows a simple rule for mixing up the characters in the plaintext to form the ciphertext.

As an example, to encrypt the message ATTACKATDAWN with the keyword KEYS, we first write out our message as shown below,

K	E	Y	S
A	T	T	A
C	K	A	T
D	A	W	N

Note: if the message to encode does not fit into the grid, you should pad the message with x's or random characters for example, ATTACKNOW with the keyword KEYS might look like below,

K	E	Y	S
A	T	T	A
C	K	N	O
W	X	X	X

Once you have constructed your table, the columns are now reordered such that the letters in the keyword are ordered alphabetically,

E	K	S	Y
T	A	A	T
K	C	T	A
A	D	N	W

The ciphertext is now read off along the columns, so in our example above, the ciphertext is TAATKCTAADNW

You should demonstrate your implementation by encrypting the file in the folder Q7 using the keyword - LOVELACE.

Q8. Starting with the number 1 and moving to the right in a clockwise direction a 5 by 5 spiral is formed as follows:

```
21 22 23 24 25
20  7  8  9 10
19  6  1  2 11
18  5  4  3 12
17 16 15 14 13
```

It can be verified that the sum of the numbers on the diagonals is 101. i.e. $17 + 5 + 1 + 9 + 25 + 21 + 7 + 3 + 13 = 101$

Write a program in Java to find the sum of the numbers on the diagonals in an X by X spiral formed in the same way, where $5 <= X <= 1001$. Demonstrate your code by printing out the sum of the numbers on the diagonals in an 1001 by 1001 spiral.

Note, this problem is inspired by Project Euler so, as stated in the rules of Project Euler, your solution should return an answer under 60 seconds.

3 Marking Scheme

Marks are distributed equally across all 8 questions. Each question will be marked out of 10, which can be broken down into:

- Correctness: The solution is a correct implementation of the question. Where appropriate, the correct output is given by the program. Note: half of these marks come from implementing your own functions/methods which replace (where appropriate) inbuilt methods/functions. (Max: 4 marks)

- Code Quality: Appropriate programming techniques have been used for each language. Code attempts to optimise the implementation of algorithms. (Max: 4 marks)

- Commenting and Formatting: Code is indented consistently and contains appropriate commenting throughout. (Max: 2 marks)

4 Submission

By the date/time specified above, you should upload a .zip file. The name of the zip file should be in the form:

CW2-yourid.zip

The zip file must contain your source code and any resource files needed - note that compiled code will not be considered. Your solution must not have any package declarations.

Failure to follow the specifications, such as by submitting non-zip files or using package declarations, may result in penalised marks.