CM10228 Coursework 2
Problem Solving in Java and C

12th April 2021

1. Introduction

In your second programming assignment for this semester, you will have to programmatically solve eight questions in both Java and C. You can use any Integrated Development Environment (IDE) for the development of your code. Your code must compile and run on the University's linux.bath.ac.uk server without requiring the installation of additional libraries, modules or other programs. The linux server features **Java 11** and any C code will be compiled using the **GNU compiler collection (gcc)**.

Please check Moodle for the submission deadline.

2. Submission

By the date/time specified on Moodle, you should upload a single zip file which contains your solutions for each attempted question. **You must download the templates (CM10228-CW2.zip) from the Moodle submission link and answer each question according to the given method/function signatures.** You may add additional functions/methods, and/or classes, however the files must be able to run according to the instructions in the templates. **The method/function signatures provided for each question should not be modified**. **We will run automated tests based on the given method/function signatures (i.e. variations of the code present in the main methods of the templates), if you fail these you will NOT be awarded any marks for correctness.** You may change the code in the main methods/functions (e.g. to include different examples to test), and additional files are present for you to be able to test your solution (where appropriate).

The template files containing your answers should be placed in their respective directories (as downloaded from Moodle) which should be in the root of the zip file. The name of the zip file must be **CM10228-CW2.zip**. Please do not include any identifying information. **Failure to follow the submission specifications, by providing unneeded files or not following the .zip structure specified, will result in a penalty being applied to the final mark.** Your .zip file must be submitted to the Moodle assignment page by the submission deadline. Submissions received after this deadline will be capped at 40% if received within 5 working days. Any submissions received after 5 working days will marked at 0%. If you have a valid reason for an extension, you must submit an extension request through your Director of Studies – unit leaders cannot grant extensions.

3. Problem Questions

Question 1 (Java):

Implement the Shellsort algorithm (https://en.wikipedia.org/wiki/Shellsort) for an array of up to 1000 signed doubles in Java. Your solution must use concrete gaps of [1, 3, 7, 15, 31, 63, 127, 255, 511]. Your solution must print the (partially) sorted array after each gap on a new line in the form:

`[`a_0`, `a_1`, `a_2`, …, `a_n`]`

Where a_n is the nth element in the (partially) sorted array (please note the space after the commas), and each element should be formatted to 2 decimal places (e.g. 1.00).

Question 2 (Java):

Implement interpolation search for a list of Strings in Java. The method should return the position in the array if the string is present, or -1 if it is not present.

Question 3 (C):

Implement your own Hash Table in C for storing and searching names, i.e. char arrays. Each name stored in the hash table should be unique. In the event of collisions, you should use linear probing with an interval of 1. Your Hash Table implementation should have the following interface:

`int hash_function(const char *key)` – the hash function should be the sum of the ASCII values of the *key* string modulo the size of the underlying data structure.

`void resize_map(int new_size)` – resizes the underlying data structure. The hash map should also resize when the load factor is greater than 0.7.

`void add_to_map(const char *name)` – adds the string to the hash map.

`int remove_from_map(const char *name)` – removes the string from the hash map if it is present. Returns true for success, false if the string can not be found.

`int search_map(const char *name)` – searches the hash map for the given string. Returns true for success, false if the string can not be found.

`void print_map()` – prints the hash map (you may choose how you wish to format this).

Questions continue over page…

Question 4 (C):

Implement your own XOR Linked List (https://en.wikipedia.org/wiki/XOR_linked_list) in C capable of storing names. Strings (including null terminator) should not be added to the linked list if they exceed 64 bytes in length. Your implementation should have the following functions:

`void insert_string(const char* newObj)` – Inserts the string at the beginning of the XOR linked list.

`int insert_before(const char* before, const char* newObj)` – If possible, inserts before the string "before" and returns true. Returns false if not possible (e.g., the before string is not in the list).

`int insert_after(const char* after, const char* newObj)` – If possible, inserts after the string "after" and returns true. Returns false if not possible (e.g., the after string is not in the list).

`int remove_string(char* result)` – If possible removes the string at the beginning of the XOR Linked list and returns its value in result. If successful return true, otherwise returns false.

`int remove_after(const char *after, char *result)` – If possible, removes the string after the string "after" and fills in the result buffer with its value. If successful return true, otherwise returns false.

`int remove_before(const char *before, char *result)` – If possible, removes the string before the string "before" and fills in the result buffer with its value. If successful return true, otherwise returns false.

`void print_list()` – Prints the contents of the list in order.

Question 5 (C):

Implement an algorithm in C which given a file containing a block of text as input, redacts all words from a given set of "redactable" words (also from a file), and outputs the result to a file called "result.txt". 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 in "result.txt" 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 must not use any of the string libraries to answer this question – if you do you will lose all marks for correctness.** You may develop your own versions of the functions in these libraries (e.g. strlen, strcmp). You should also test your program using the example files provided.

Questions continue over page…

Question 6 (Java):

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 and outputs the result to a file called "result.txt". 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.

Questions continue over page…

Question 7 (C):

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 ATTACK AT DAWN with the keyword KEYS, we first write out our message as shown below (note we remove the spaces):

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, ATTACK NOW 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 can test your implementation by encrypting the file in the folder Q7 using the keyword - LOVELACE.

Question 8 (Java):

You are given the following information, but you may prefer to do some research for yourself.

- 1 Jan 1900 was a Monday.
- Thirty days has September, April, June and November. All the rest have thirty-one, saving February alone, which has twenty-eight, rain or shine. And on leap years, twenty-nine.
- A leap year occurs on any year evenly divisible by 4, but not on a century unless it is divisible by 400.

How many Tuesdays fell on the first of the month during the twentieth century (1 Jan 1901 to 31 Dec 2000)?

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.

4. 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.
  (Max: 6 marks)

- Code Quality: Appropriate programming techniques have been used for each language. Code attempts to optimise the implementation of algorithms.
  (Max: 2 marks)

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