UNIVERSITY OF MALTA
FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

CPS1011 Programming Principles (in C)

Assignment 2017/8

Instructions:

1. This is an **individual assignment**.
2. You may be required to demonstrate and present your work to an exam board.
3. The hard copy of the report must be handed in to the Computer Science Dept. clerk's office by the assignment deadline. A soft-copy of the same report along with all digital artefacts must be uploaded to the VLE upload area by the same deadline. All files must be archived into a single .zip file. It is the student's responsibility to ensure that the uploaded zip file and all contents are valid. You must include all source, make files, any scripts and instructions required to compile the code.
4. Reports (and code) that are difficult to follow due to low quality in the writing-style/organisation/presentation will be penalised.
5. Assignment queries are to be made strictly during the beginning of lectures or posted to the assignment's forum on VLE, as of when individual tasks are announced in class till one week before the deadline (excluding recess).
6. This assignment comprises 100% of the final CPS1011 assessment mark.
7. You are to allocate 50 to 60 hours for this assignment.
8. The report must be organized according to the task sequence, and for each clearly explaining the relevant code fragments as well as the program's output listings. The full source code files must also be uploaded to VLE (as explained above) along with 2 `CMakeLists.txt`, one for each of tasks 1 and 2. A `readme.txt` file must be included in the root directory of the archive file describing the content's organization.

9. While it is strongly recommended that you start working on the tasks as soon as they are announced in class, the firm submission deadline is **Monday 15th January 2018**.

1. **Problem solving.** (Total-50 marks)

   Tasks :-

   (a) Tom invests €200 at 15% simple interest. (That is, every year, the investment earns an interest equal to 15% of the original investment.) Joan invests €200 at 10% interest compounded annually. (That is, interest is 10% of the current balance, including previous addition of interest.) Write a program that finds how many years it takes for Joan's invested sum to overtake Tom's along with a display showing the two values at that time.

   [10 marks]

   (b) *YourGreens.com* sells artichokes for €2.05/kg, onions for €1.15/kg, and carrots for €1.09/kg. It gives a 5% discount for orders of €100 or more prior to adding shipping costs. It charges €6.50 shipping and handling for any order of 5 kg or under, €14.00 shipping and handling for orders over 5 kg and under 20 kg, and €14.00 plus €0.50 Eur/kg for shipments of 20 kg or more. Write a program that repeatedly prompts users to enter the next amount of vegetables such that: a response of `a` lets the user enter the kg of artichokes desired, `b` the kg of onions, `c` the kg of carrots, and `q` allows the user to check out. The program should keep track of cumulative totals. That is, if the user enters 4 kg of onions and later enters 5 kg of onions, the program should report 9 kg of onions as a single bill item.

   Ultimately the program displays an itemized bill, including a breakdown of discounts and shipping costs. The program should be written in a way so that only the final itemized bill output can be redirected to a text file. User input validation is a requirement.

   [10 marks]

   (c) Write a program that reads in an input text file and differentiates between C source files (typically include a `#include` directive towards the beginning and are case sensitive), HTML files (typically include the case-insensitive `<html>` and `</html>` tags towards the beginning and end respectively), and 'other type' of files. The program is permitted a margin of imprecision.

   [10 marks]

(d) Write a program that reads an input text file, identifying and auto-correcting the following common typing errors: 'Space before comma or a full-stop', 'Multiple spaces as opposed to a single space', and 'Missing spaces'. Detection of the last typo makes use of the heuristic that 'words longer than 12 characters and not including a hyphen (-) are rare'. The user must be prompted to confirm auto-correction just in these cases.

[10 marks]

(e) Write a program that includes a memory introspection function `view_stack_frame()`, printing out the stack frame of caller functions. For simplicity's sake, you may assume that all variables inside the callers' stack frames are all of the same type, although their number may vary. The function displays output organized in two columns, the first displaying addresses of variables while the second displays their corresponding values. Symbolic identifiers do not feature.

[10 marks]

2. **Hash tables.** (Total-50 marks)

A hash table is a data structure central to a number of search algorithms and API object implementation in high-level languages e.g. Python's dictionary and Java's `java.util.HashMap<K,V>`. Hash tables provide a concrete way with which information (*values*) stored along a corresponding *key*, which is subsequently used to search and retrieve the associated information at a later stage. While variations abound, one approach to a hash table is to combine an array's direct access facility serving as an index, with the flexibility (but sequential access constrained) of linked lists used to store values as in the following depiction:

where given a key-value pair as input (character-integer in this case), a 'hash' is computed out of the key (e.g. ASCII index `mod` 5), with a new hash table node being created and placed on the linked list associated with the computed hash serving as entry into the array index. In this case the hash table is said to support collisions, since multiple keys map to the same hash. In general keys are desired to be unique. Key-value pair retrieval takes a key as input, and following hash computation a

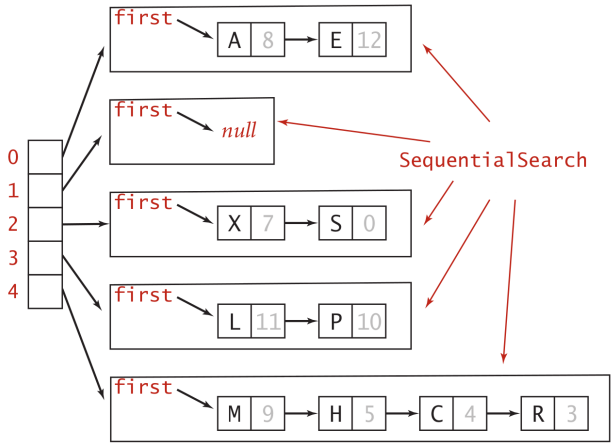

Figure 1: A hash table implementation using an array for direct access of the computed hashes, with linked lists storing key-value pairs with collisions.

sequential search of the corresponding linked list ensues until a match is found or `null` is encountered.

In this task you are required to incrementally build towards a hash table on the lines of the one just described, ultimately having the following characteristics :-

- Keys and values are both character strings;
- Hash table creation supports a parameterizable hash space, and therefore the hash computation is parameterized accordingly;
- Insertion, deletion and look-up of key-value pairs are supported,
- as well as data structure save/load to/from disk.

You are to provide three versions of the implementation, each version serving as an increment leading to the final implementation according to the level of sophistication provided:

(a) A fixed 2D array version, where the number of rows is equal to the hash space, while columns cater for key-value pair storage. In this case the hash space is pre-fixed, as well as the maximum possible number of possible collisions.

[10 marks]

(b) A dynamic 2D array version of the above, where this time the hash space is fully parameterizable and the number of collisions is solely bounded by program memory availability.

[10 marks]

(c) The final linked list version exactly as depicted in Figure 1.

[15 marks]

(d) Provide a single test driver program (no user input is required) to test all versions of the hash table. All versions must be compiled as shared libraries implementing the 'same' Abstract Data Type (ADT) API as exposed by a single header (`.h`) file.
[Shared library compilation and linking - 5 marks; API - 5 marks; Test driver - 5 marks]

One example use case for the resulting library is that of a stream editing application. It substitutes any strings corresponding to dictionary keys found within the input stream with their corresponding values within the output stream, in an efficient manner. You may want to try this out in your free time.

---

end of assignment