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

CPS1011 Programming Principles (in C)

Assignment 2023/24

Instructions:

1. This is an **individual assignment**.
2. You may be required to demonstrate and present your work to an exam board.
3. A soft-copy of the report in pdf format must be uploaded to the VLE upload area, along with the code repository, archived in a single `.zip` file by the deadline given below. 2 `CMakeLists.txt` files, one for each of tasks 1 and 2, are to be provided. A `README.md` file must be included in the root directory of the repo describing the content's organization.
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 and output snippets. The full source code files for each task must also be privately shared on gitlab with mmarrkv set as a developer collaborator.
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 **19th January 2024 at NOON**.

1. **Algorithm implementation.** (Total-45 marks)

   (a) Research and implement the Secant root-finding method. It is an iterative algorithm which computes the successive root approximation x_n as follows:$x_n = x_{n-1} - f(x_{n-1})(\frac{x_{n-1}-x_{n-2}}{f(x_{n-1})-f(x_{n-2})})$.

   [15 marks]

   (b) Research and implement the Newton-Raphson root-finding method. It is also an iterative algorithm which computes the successive root approximation x_n as follows:$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$.

   [15 marks]

   For both the above tasks, (a) and (b), you are required to implement each algorithm in its own C function, and only for polynomial single-valued functions $f(x)$, where x is a scalar and $f(x)$ is at most degree 5. The polynomial function must be passed as an argument in each case, parsed from user input and transformed into a form that each C function can process.

   (c) Write a program that presents the end-user with a command-line menu, repeatedly asks the user to execute the above functions, and displays their output accordingly, or else to quit. Proper validation of user input is required.

   [15 marks]

2. **A Generic Set library.** (Total-55 marks)

   In computer science, *sets* are typically used as a simple collection of objects, called elements, that contain no duplicates. In this task, you are to implement a library that provides an implementation for a *Generic Set*, meaning *it can be instantiated for multiple types*, but with each instance being only concerned with a single element type per set instance. The result is a new data type called `GenSet_t` supporting the following operations:

   - `initSet()` - Creates a new set for a specific element type and allocates associated memory resources.
   - `deinitSet()` - Destroys an existing set and relinquishes associated memory resources.
   - `addToSet()` - Adds a non-duplicate element to the set.

- `displaySet()` - Outputs all elements of a set in no particular order to the standard output.
- `unionSet()` - Returns a newly created set, the union of two input sets.
- `intersectSet()` - Returns a newly created set, the intersection of two input sets.
- `diffSet()` - Returns a newly created set, the difference between two input sets.
- `countSet()` - Returns the element count, or cardinality, of an input set.
- `isSubsetSet()` - Returns whether the first set is a subset of the second.
- `isEmptySet()` - Returns whether a set is empty.

All operations above refer to the usual set operations. Functions that take more than one set as input must ensure that *all input sets are instantiated for objects of the same type*. Set elements are to be stored in dynamic memory (on the heap).

Tasks: -

(a) Implement a simplified version of `GenSet_t` that is instantiated with either elements of integer type or of fixed-sized 64-character strings. Both options need to be provided. Test all functionality.

[20 marks]

(b) Extend the implementation in 2(a) so that there are no limitations on the supported element types this time. Additionally, support for the following additional operation is required:

`export()` - Exports the Generic Set elements to a text file, with each line displaying a textual representation of each element in no particular order.

Test the additional functionality.

[20 marks]

(c) Compile the full implementation as a shared library and provide a proper Abstract Data Type interface. The test application should link with the shared library. In case of difficulties with task 2(b), you can compile the library from the simpler version developed in 2(a), with a maximum of 10 marks.

[15 marks]
