SOFT 153

Algorithms, Data Structures, and Mathematics

COURSEWORK -- Part A (Algorithms)

The SOFT153 coursework assignment is made up of two (equally weighted) parts:

- **Part A (Algorithms and data structures) – Because we may wish to run your code, the submission for Part A is via the DLE.**
- **Part B (Mathematics). The specification for this part of the coursework will be posted elsewhere on the SOFT153 DLE site. Type-setting mathematics is extremely time-consuming and tiresome: to save you this effort we are allowing you to provide a hand-written submission for Part B; submission for Part B is therefore hard-copy via the Faculty Office (ground-floor Smeaton).**

Both parts of the coursework must be submitted by 10am, Friday 13th May

Module Leader: Thomas Wennekers
thomas.wennekers@plymouth.ac.uk
School of Computing, Electronics and Mathematics

This assignment is to be done individually and is worth 50% of the coursework mark.

Format of coursework to be handed in:

Provide **a single C#** file containing the functions/methods/sub-routines requested in the task defined below and a Main program that tests the routines appropriately. The file should be *suitably laid out and commented according to good programming style*.

Don't code graphical user interfaces -- use a console application. GUIs will not attract more marks, but likely make the code less readable. This could reduce marks.

Don't use programming constructs on top of the basic "C-core".

Keep your code simple and clear.

Late assignments policy

You are reminded about the University's policy for late assignments. You can submit for up to 24 hours after the hand-in date, but in this case the assignment can only receive a maximum of pass-level marks (40%). Later assignments will automatically be awarded zero. You are, therefore, encouraged to submit the assignments prior to the deadlines. The DLE system allows to replace previously submitted files as long as the hand-in date has not passed.

Plagiarism:

Students are warned that the University takes plagiarism very seriously. You are advised to read the section in your student handbook on Examination and Assessment Offences. As a brief guide, plagiarism is the deliberate use of another person's work without attribution or acknowledgement. DO NOT construct your assignments from code taken from another student or from the Internet. There is no objection to you discussing your solution with other people, but the code you hand in must be your own.

This assignment assesses the following learning outcomes:

1. Recognise and explain the importance of algorithmic design in optimising use of computing resources.
2. Identify suitable structures and algorithms to implement programming tasks.
3. Synthesize the solution to a real-world task as a combination of two or more standard algorithms

Please read carefully the specifications for the tasks given below – you will lose marks if your programs do not do what is specified.

Write a console program to allow a user to do conversions between two different units of measurement. So, the user might type in an amount, and the names of two units of measurement, for example (note the format):

5, ounce, gram

In this case the program should then print out the number of grams equal to 5 ounces, thus:

5 ounces is 140 grams (1 ounce is 28 grams)

Unit names and conversion factors will be supplied as a file convert.txt, and this should be read in once by the program at start-up. The file consists of a series of lines, each consisting of two unit names (both strings) and a conversion factor (a number with a decimal point) separated by commas, for example:

ounce,gram,28.0

This means that one ounce equals 28.0 grams. There may be blanks (as in "gram , ounce, 0.036") A sample file is available on the portal, but your program should be able to cope with **any** file laid out in this way. You will need “using System.IO;” at the start of the program for functions that open and read from a file. You are allowed to use a StreamReader and the corresponding method .ReadLine (check google or consult a textbook about C#). .Split and .Length are allowed, .ToDouble, .ToInt, .ToString are allowed. If you need more help with how to proceed with this program, see your practical class supervisor.

If the user types in an incompatible pair of units (such as 5, gram, pint) the program should take appropriate action. Your program should repeatedly prompt the user for an amount. As long as a positive amount is typed in, the program should proceed with the conversion; when a zero or negative amount is typed in the program should halt.

Do also read the general requirements and comments on pages 1 to 3 of this document. They are part of the assignment specification.

Marking Scheme

Marks will be provided according to the following main categories

1. General Code properties (25%)
   This category refers mainly to good programming style. More specifically, aspects considered will be: Proper modularisation of the algorithmic structure; the correct use of appropriate data structures; originality of the solution; correctness and completeness; appropriate names of identifiers and methods; code readability; consistent lay-out and indentation; good commenting (don't comment the obvious, but do comment all crucial steps); any error handling. Also remember, that C# instrinsic data structures like Lists or ArrayLists should not be used.
2. Task Specific Code Properties (75%)

This refers to how the problem to implement the task is solved, and whether all features asked for in the task specification have been implemented. How the main data structures and algorithms have been implemented is also important. There are various different possibilities to do this, some basic, some more advanced and more universal. Better solutions will receive higher marks. Code efficiency with respect to memory space and execution time is another aspect here.