CM20214/CM20221
Advanced Programming Principles/Programming II
Assessed Coursework Assignment 1

Set:	Friday 8 Nov 2013
Due:	5pm., Monday 6 Jan 2014, via Moodle
Marks:	10% for CM20214, 20% for CM20221 of course total
Set by:	Russell Bradford
Environment:	Lisp

This coursework is to exercise your ability to **think and program in the functional style**.

Implement simple polynomial arithmetic in Lisp. This should be able to do calculations like $(x + y)(x + y) \rightarrow x^2 + 2xy + y^2$, with the polynomials represented in some suitable way within Lisp.

Your code should

- implement the arithmetic operations $+$, $-$ and $*$. The functions should be named `poly+`, `poly-` and `poly*`
- expand and collect together like terms, e.g., the sum of $x + y$ and x should be $2x + y$ rather than $x + y + x$; while the product of $(x + y)$ and $(x + z)$ should be $x^2 + xz + xy + yz$. The order of the terms is your choice.

This will involve you picking a suitable representation for your polynomials (hint: Lisp has symbols and lists!), and then implementing the required functions to manipulate them.

- `poly+`, `poly-`, `poly*` should all be functions of exactly two arguments, both being polynomials presented in your chosen format, and should return a polynomial in your chosen format. Thus, if `p1` is $x + y + 1$ and `p2` is $2xy + x + z$, then `(poly+ p1 p2)` returns your representation for $2xy + 2x + y + z + 1$; and `(poly- (poly+ p1 p2) p2)` returns your representation for $x + y + 1$.

All code must be in the functional style.

CM20221 only

Additionally to everything above, you should

- implement the above arithmetic (i.e., $+$, $-$ and $*$) for *Laurent* polynomials, namely having negative powers, e.g., $x^2 + z^{-1} + x^{-1}y$.
- Do differentiation on Laurent polynomials, e.g., $x^{-2} + y + 1 \rightarrow -2x^{-3}$ when differentiating with respect to x. The function should be called `polydiff`).
- `polydiff` should a function of exactly two arguments, the first a polynomial, the second a variable to differentiate with respect to. It should return a polynomial.
- Provide a short essay describing the issues that arise when you try to add division to your system. You need not implement this, but you might want to provide fragments of code to illustrate points.

Handing In

You should hand in to Moodle

- the source code listing as a file that the examiners can load into a Lisp. Please make it clear which Lisp you used!
- a substantial number of examples that show you have tested your programs thoroughly
- the results of the testing
- CM20221: and the essay, as plain text or a PDF
- all in a single zip file, named `yoursurname-youruserid.zip`, e.g., `bradford-masrjb.zip`

Testing will be worth a significant number of the assignment marks.

Also include any information that you feel will help the examiners to understand your program, for example, descriptions of the datastructures you use; descriptions of algorithms you use; which Lisp you have used; and so on.

Make sure you include the results of your testing: cut and paste the results into the documentation (screendumps are often illegible) or supply as a separate file.

Feedback

Feedback on this assignment will be provided via Moodle and a problems class after the hand-in, if requested.

Notes

- A polynomial is a sum of terms; a term is a constant coefficient multiplying product of variables raised to integer powers.
- Do not spend time on reading and pretty printing algebraic expressions, but do all input and output in your chosen Lisp form. In particular, outputs from your functions can be used as inputs to your functions.
- There is no need for validation of inputs—you may assume all inputs to your `poly` functions are valid.
- CM20221: general rational terms like $1/(x+1)$ are not required, merely sums of products of powers of variables.
- Your choice of polynomial representation is very important and will strongly affect the complexity and amount of code you will have to write
- You are recommended to stick to Clisp or Euscheme or Xlisp, but any classical-style Lisp is acceptable: ask if you are unsure about a particular Lisp.
- Non-functional style operators include (but is not limited to) `set`, `setf`, `setq`, `push`, `pop`, `mapcan`, `nconc` and some forms of `sort`. Use of non-functional code will lose marks. Thus code with side effects (e.g., using global variables, printing values rather than simply returning them, etc.) will lose marks.
- Create your code using a normal text editor (not a word processor!); then load it into your Lisp using `load` or similar.
- Keep the code listing to a maximum of 80 columns width: super-wide lines are harder to read and tries to put too much into a single line.
- Avoid the Common Lisp `loop` construct: it's over-complicated and you can usually achieve what you want in a simpler way.
- Your testing should be structured and designed to test all pathways through your code.
- The accompanying material (e.g., the essay, descriptions) should be either plain text or PDF, *not* DOC, DOCX or other variant.
- This is individual coursework and all work must be your own.
- Check Your handin was successful. Moodle has been known to lose coursework.
- This assignment will be much easier if you have invested time in doing the simple Lisp exercises.
- You are not expected to use classes to implement the coursework; simple datastructures, e.g., lists, will suffice.
- **Don't leave this assignment until the last moment before starting: that is a sure route to disaster**.

All answers must be submitted to Moodle by 5 p.m. on Monday 6 Jan 2014.

Assessment Criteria

- Normal standards for coding and commenting apply
- High marks will be gained by versions that are extensively tested and work on a wide variety of examples. CM20221: and have a well-reasoned essay
- Medium marks will be gained by versions that work mostly or have a reasonable amount of testing. CM20221: or have a weak essay that misses a few important points
- Low to failing marks will be gained by versions that don't work or have limited functionality (e.g., single variable polynomials only) or have limited testing, particularly of the edge cases, or employ non functional style code. CM20221: or have a poor essay that misses the important points

Applications for extensions to the hand-in date must be made to the Director of Teaching, John Power.