625-600: Programming Assignment 1

Read every page very carefully before you begin.

1. Implement `deriv` to support:

   addition, subtraction, unary minus, multiplication, and division.
   → HINT: use slide02 page 44 as a skeleton.

2. Implement simplification routines `splus` etc. for all operators and
   integrate it into `derivplus`, etc.
   → HINT: Integrate code in slide02 page 45 into code in page 44. (Code
   available on course web page, under the src/ directory.)

3. Write a simple function `deriv-eval` to assign a numerical value to the
   variable and get a single number corresponding to the resulting derivative:
   `(deriv-eval '(+ (* x x) (- 2 x)) 'x 20)`
   * You can either use recursion or some other neat lisp tricks to achieve this.

Programming Assignment 1: other conditions

1. Use only one variable (say `X`). Other symbols should be treated
   as constants (e.g. `Y, Z, ...`).

2. All operators should be binary operators:
   i.e. expressions like `(+ 1 2 3 4 5)` do not need to be
   supported. Only those in the form of `(+ 1 2)` are expected to
   be used.

3. The only exception is the unary minus operator `(- 10)`, which
   only has one argument.

4. You must check for division by zero and print an error message in
   case such an event occurs.

Programming Assignment 1: Example Inputs and Outputs

1. `(deriv '(* (+ x 4) (+ x 5)) 'x)`
   `-> (+ (+ X 4) (+ X 5)))`

2. `(deriv '(/ (+ x 1) x) 'x)`
   `-> (/ (- X (+ X 1)) (* X X))`

3. `(deriv-eval '(* (+ x 4) (+ x 5)) 'x 10)`
   `-> 29`

4. `(deriv-eval '(/ (+ x 1) x) 'x 5)`
   `-> -1/25`

Programming Assignment 1: Required Material

Use the exact filename as shown below (in **bold**).

- Program code (**deriv.lsp**): put it in a single text file.
  – Ample indentation and documentation is **required**.

- Documentation (**README**): user manual (how to load and
  execute).

- Sample inputs and outputs (include in **README**)
  – 5 non-trivial examples, each containing a combination of more
  than 5 arithmetic operators. Provide examples for both `deriv`
  and `deriv-eval`.

- Grading criteria:
  – README, test cases, comments, readability: 10%
  – deriv : 50%
  – simplification: 40%

Programming Assignment 1: Important Grading Information

- Since the deriv functions call the simplification functions, if the simplification routine is broken, regardless of the deriv functions being correct, your call will result in an error. If this happens, **both** deriv and simplification will be graded as malfunctioning.

- If you got deriv functions to work, but if simplification is not working, take out the simplification code from your deriv functions so that at least your deriv functions work.

Programming Assignment 1: Submission

- Use the **csnet** turnin form (http://csnet.cs.tamu.edu).

- Due : see course web page for the due date.

- Late policy: No late submission is allowed. Submit whatever you have by then.

- Only include plain text ASCII files. **Do not include MS-Word documents or other formatted text. Your assignment will not be graded in such a case.**

Academic Policy

- This is an **individual assignment**. No collaboration is allowed.

- Any suspected academic policy violations will be promptly reported to the **Aggie Honor System Office** (http://www.tamu.edu/aggiehonor/).

- For discussions regarding the assignemnt, frequently check out the **read-only bulletin board** on the course web page, since most of the Q-and-A's will be uploaded there.