CS 203 F19 Project 1: Creating A Data Structure

C W Liew Version as of: 11:49 Friday 11th October, 2019

Due: 11:59pm, October 20, 2019

Goals

The goals of this project are:

1. gain experience in designing and implementing C programs,
2. gain a better understanding of how to implement data structures in C
3. gain a better understanding of recursive structures and functions

General Description

In this project you will design and implement (in the C programming language) a data structure will allow you to manipulate expressions. The program will read in commands from the command line and manipulate the data structure and print the result.

The expressions you will read in will be of the form:

```
( <operator> <space separated list of operands> )
```

or

```
< number $>
```

Examples

Here are some examples of valid expressions (there is a recursive definition)

```
2
(+ 2 4)
(+ 2)
(+ 2 (+ 2 4) 6)
```

The valid operations are: ”+”, ”-”, ”*”.

Data Structure

The basic structure you will use consists of a Expression (Exp) defined as:

```
char* symbol or Exp* first;
Exp* rest;
```

where the symbol can be an operator (e.g., ”+”) or a number (e.g., ”256”).
So the structures corresponding to the examples would be:

- Exp(2)
- ((Exp(+) $->$ Exp(2)) $->$ Exp(4))
- (Exp(+) $->$ Exp(2))
- (Exp(+) $->$ Exp(2) $->$ (Exp(+) $->$ Exp(2) $->$ Exp(4)) $->$ Exp(6))

Commands

The commands that can be used are:

- 'c': create a new expression. The next line contains the expression. A new data structure is created and the current expression will point to it.
- 'p': print the current expression
- 'e': evaluate the current expression (arithmetically)
- 'a': append a new expression to the current expression. The new expression is given on the next line

  Assuming the current expression is: "(+ 1 2 3)".

  Appending "4" would result in "(+ 1 2 3 4)", while appending "(+ 4)" would result in "(+ 1 2 3 (+ 4))"
- 's': prints a subset of the expression. The subset specification is on the next line and uses 'f' (first) and 'r' (rest)

  For exmaple, if the current expression is "(+ 2 (- 3 4) 5)"

  - "f" results in "+"
  - "r f" results in "2"
  - "f f" does not make sense
  - "r r f" results in "(- 3 4)"

Example

```
c
(+ 2 4 (- 10 7) (* 3 (+ 2 5)))
p
(+ 2 4 (- 10 7) (* 3 (+ 2 5)))
e
30
```

```
a
10
p
(+ 2 4 (- 10 7) (* 3 (+ 2 5)) 10)
e
40
s
r r r f
(-10 7)
```

Figure 1 shows you the structure corresponding to the expression:

```
(+ 2 4 (- 10 7) (* 3 (+ 2 5)) 10)
```

Grading

Your program will be graded on the following criteria:

- functionality - how much of the specified functionality works?
  - basic - create, print an expression
  - good functionality: evaluate
  - very good functionality: append
  - max functionality: subset
- design - is your program decomposed in an appropriate manner?
- documentation - how well commented is the program

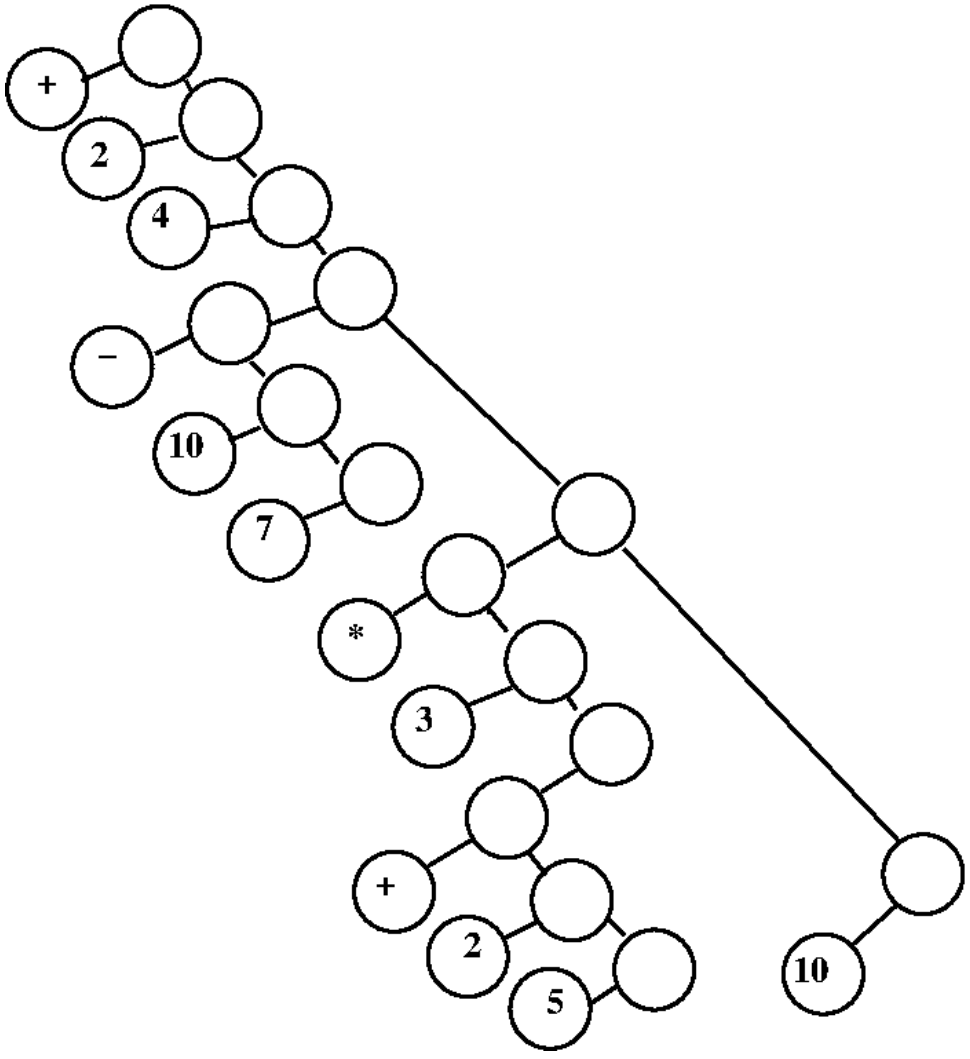

Figure 1: The structure corresponding to the example expression