# Task

Write a program in any language that can parse and evaluate any concrete syntax that complies with this specification. You can assume that the syntax will always be correct so you do not have to check for syntax errors.

# Specification

## Syntax

The­­­­­­­­­­ Tiny Programming Language is a practically useless programming language that consists of nested expressions enclosed by parentheses. A program can be evaluated by evaluating the inner most expression and working outward to a single value.

## Constants Expressions

### Number

Number can be used to represent any whole numerical value.

|  |
| --- |
| Syntax |
| (10) |

### Boolean

Boolean represents a value that can be true or false.

|  |
| --- |
| Syntax |
| (true) |
| (false) |

## Operators Expressions

### Plus

The plus operator can add two numbers together.

|  |
| --- |
| Syntax |
| (+(1)(1)) |

### Minus

The minus operator can subtract one number from another. The second number expression is subtracted from the first number expression.

|  |
| --- |
| Syntax |
| (-(1)(1)) |

### Equal

Two expressions are considered equal if they evaluate to the same value.

|  |
| --- |
| Syntax |
| (=(1)(1)) |

### Ternary

The ternary operator can be used to conditionally evaluate an expression. It takes 3 expressions where the first must evaluate to a Boolean value and can be read as ‘if <condition> then <first> else <second>’

|  |
| --- |
| Syntax |
| (if  (true)  (15)  (16)) |

## Function Expressions

### Sum

The sum expression can calculate the total of multiple numbers. It takes the form of sum followed by any number of expressions that evaluate to numbers.

|  |
| --- |
| Syntax |
| (sum(1)(2)(3)) |

# Examples of reducing expressions

|  |
| --- |
| (5)  => (5) |
| (true)  => (true) |
| (false)  => (false) |
| (+(5)(6))  => (11) |
| (-(15)(6))  => (9) |
| (-  (+(10)(5))  (-(8)(2)))  =>  (-(15)(6))  => (9) |
| (=  (+(1)(1))  (1))  =>  (= (2)(1))  => (false) |
| (if  (true)  (15)  (16))  => (15) |
| (if  (=(1)(2))  (15)  (16))  =>  (if  (false)  (15)  (16))  => (16) |
| (sum(1)(2)(3))  => (6) |
| (sum  (-(5)(2))  (3))  =>  (sum  (3)  (3))  => (6) |
| (sum  (-  (if  (=(1)(2))  (15)  (16))  (2))  (-(5)(2))  (sum(2)(2)(2)))  =>  (sum  (-  (if  (false)  (15)  (16))  (2))  (-(5)(2))  (sum(2)(2)(2)))  =>  (sum  (-(16)(2))  (-(5)(2))  (sum(2)(2)(2)))  =>  (sum  (14)  (3)  (6))  => (23) |

# Marking

Handle any number => 5

Handle bools => 5

Handle equality => 10

Handle the + operator => 10

Handle the – operator => 10

Handle nested expressions (e.g. "(+(-(3)(12))(6))") => 20

Handle conditional => 20

Handle sum => 20

**Submission**

Your submission is due by 4:00pm today. Any programming/scripting language can be used. You should create and zip a directory containing your solution (all code and executables (If code compiles to an executable)). Please email this to me by the deadline at: shane.brady@sig.com

Your application/script should accept one command-line parameter, which is an absolute path to a file, and should output the result to the console.

I will go through your code to allocate marks. I will test your solution with various data files, something like this:

tpl.exe C:\code.txt

Any of the examples from this document can be used for testing.