

Program 2^x

More Scheme practice for extra credit.

Using **boost.org**'s λ -library for C++ as an inspiration, let's take it back to Scheme by writing a more flexible "implicit lambda". For example, take the function:

$$f(x_1, x_2) = \frac{1 + x_1}{1 - \sqrt{x_2}}$$

In ordinary Scheme, we could define f this way¹:

```
(define f
  (lambda x
    (/ (+ 1 (car x))
       (- 1 (sqrt (cadr x))))))
```

Write a Scheme procedure **ilambda** that allows you to use this (*arguably*) clearer syntax:

```
(define f
  (ilambda '(/ (+ 1 _arg1)
              (- 1 (sqrt _arg2)))))
```

You are writing a procedure, not a syntactic form, so note the quote. Your procedure should use the special symbols **_arg1** through **_arg4** to implicitly bind arguments. You do not need to check for syntactic errors.

One handy function for this assignment is **gensym**. This is a no-argument function that returns a symbol with a new, automatically generated name.

As always, keep your code clean and simple. A few auxiliary functions will be helpful. Submit your work in the file **ilambda.scm**.

Due: By email, Friday December 13 at noon. This is an individual assignment.

¹ Note we are using **(lambda x ...)**, as opposed to **(lambda (x)...) .** The latter is a function of one argument. The former is a function of any number of arguments; the *list* of arguments is bound to **x**.