Rippl

Recursively Inferred Pure functional Programming Language

Final Report

Riddler: Da Hua Chen

Gallbladder: Hollis Lehv

Language Yoda: Amanda Liu

Prime Minister: Hans Montero

{dc2802, hml2138, al3623, hjm2133}@columbia.edu

Contents

1 Introduction

Rippl (Recursively Inferred Pure-functional programming Language) is a functional language that leverages the safety and elegance of pure languages like Haskell into a concise and powerful set of core language constructs. As the name suggests, Rippl is a pure language and thus does not have any mutable state and expressions have no side-effects. The language supports first-class functions, partial application, immutability, a strong static type system, lazy semantics, and list comprehensions. The unsugared syntax of Rippl is designed to more closely resemble the lambda calculus. This will hopefully give users powerful language constructs that are not only able to concisely express pure calculations, but also provide an intuitive introduction to the lambda calculus.

2 Tutorial

2.1 Setting Up Your Environment

Before you can use Rippl, please see the section on Software Development Environment under Project Plan.

2.2 Downloading and Building Rippl

Clone the repository at http://github.com/al3623/rippl.git into your home directory. Go into the source directory and run `make`. In order to be able to run the compiler in any directory, add `~/rippl/src` into your `PATH` environment variable.

2.3 Writing and Compiling a Simple Program (hollis.rpl)

Follow the commands below to run your first program in Rippl. The steps involve opening up a text file, pasting in sample code, compiling the code using the rippl compiler, and running the executable.

```
$~ vim hollis.rpl

divides = fun x -> fun y -> if (x mod y == 0)
    then true
    else false

is_prime = fun n -> if n < 2
    then false
    else
        let lst = [2...n] in
        let d = [ z | z over lst, (divides~n~z) ] in
        if head d == n
            then true
            else false

hollis = fun x ->
    let n = 0 in
    let m = 6 in
    let d = [a+b | a over [n...m], b over[1...x], is_prime~(a+b) ]
    in d

main :: [int]
main = hollis~5

$~ rippl hollis.rpl
$~ ./rippl
```

If you run the above commands, you should see:

```
$~ [2, 3, 5, 2, 3, 5, 3, 5, 7, 5, 7, 5, 7, 7, 7, 11]
```

2.4 Compiler Flags

If you want to see the program AST, you can run the compiler with the '-p' flag. This will print the AST as parsed onto the terminal screen.

```
$~ rippl hollis.rpl -p
```

If you want to see the lifted program AST, you can run the compiler with the '-l' flag. This will print the AST after lambda lifting was performed.

```
$~ rippl hollis.rpl -l
```

If you want to see the inferred types of the function definitions in your program, you can run the compiler with the '-t' flag.

```
$~ rippl hollis.rpl -t
```

3 Language Reference Manual

3.1 Overview

Rippl (Recursively Inferred Pure-functional programming Language) is a functional language that leverages the safety and elegance of pure languages like Haskell into a concise and powerful set of core language constructs. With list comprehensions, lazily evaluated infinite lists, a strong static type system implementing Hindley-Milner style inference, higher-order functions, Rippl sets out to provide a clean and intuitive functional programming experience like no other language can.

3.2 Type System

Rippl has primitive types, higher-order types, and arrow types.

3.2.1 Primitive Types

The primitive types in Rippl are `int`, `float`, `bool`, and `char`.

3.2.1.1 Integers

`int` is the signed integer type. An integer literal may be preceded by a unary minus (`-`) to denote a negative integer. Unary plus (+) is not supported. Integers are written in base 10 (decimal) and cannot be written in other bases (e.g. binary, octal, hexadecimal).

3.2.1.2 Floating-point Numbers

`float` is the floating-point numerical type. A float literal may be preceded by a unary minus (`-`). Unary plus (+) is not supported. Float literals must be written with a decimal point and at least one digit to the left and right of the decimal point (e.g. `-4.2`, `10.0`). This format was chosen to avoid ambiguity with integer literals when used with the list range operator (`...`). Numbers not written in this format, e.g. `6.`, `.53`, and `-.0`, are not valid floats. Floating-point numbers written in scientific notation, e.g. `1.37e15`, are also not supported. `int_to_float` is provided as a language primitive that converts integers into floats.

3.2.1.3 Booleans

`bool` is the Boolean type. There are two Boolean literals, `true` and `false`.

3.2.1.4 Characters

`char` is the character type. A character literal is written as a single character in single quotes e.g. `'A'`, `'!'`, `'+'`. Special characters, such as the newline character '\n', are escaped with a backslash (\).

3.2.1.5 Summary of Primitive Types

Type	*Size*	*Examples*
`int`	4 bytes	`0, 37, -2, 2019`
`float`	8 bytes	`0.0, -6.9, 3.1415926`
`bool`	1 byte	`true, false`
`char`	1 byte	`'#', 'x', 'D', '\n'`

3.2.2 Higher-Order Types

Higher-order types are represented as type constructors that are parametrically polymorphic in terms of type variables. The constructors for these types can be viewed as functions that take in a proper type as an argument and return a new type abstracted over the argument. The higher-order types in Rippl are lists, tuples, and the `maybe` sum type, which comprises `none` and the `just` constructor.

3.2.2.1 Lists

Lists are a first-order type and all elements of a list must have the same type. This means that a function that acts on lists need not concern itself with the type of the elements of the list. Such a function is said to be parameterized by the type of the elements in the list. All list operators in Rippl (see section 3.5.5) act on lists of arbitrary type.

Strings are handled and represented internally as lists of `char`, but string literals may be written in the usual sugared representation with double quotes e.g. `"Hello, World!"`.

Lists as well as list types are delimited by brackets, and list elements are separated by commas (e.g. `[0,3,1]`, which has type `[int]`, `['R','i','p','p','l']` ⇔ `"Rippl"`, which has type `[char]`).

3.2.2.2 Tuples

Tuples are also a first-order type, parametrically polymorphic in the types of its elements. A tuple must contain exactly two elements, but each of them can be of any type. Tuples as well as their types are delimited by parentheses, and elements are separated by commas (e.g. `("PLT",4118)`, which has type `([char],int)`).

3.2.2.3 `maybe` Type

The `maybe` type is a tool to handle errors without side effects by representing an optional value. In the case that a computation cannot return a valid value, it may return the `none` constructor of the `maybe` type. Otherwise, it can return a proper value wrapped in the `just` constructor. Because `just` can wrap any type, `maybe` is polymorphic.

3.2.3 Arrow Types

An arrow type is a sequence of types separated by an arrow (->) and represents the type of a function or operator.

```
sum_tup tup = (first tup) + (sec tup)
# sum_tup :: (int, int) -> int

apply_sec tup f = f (sec tup)
# apply_sec :: (a, b) -> b -> c -> c
```

The definition of the `sum_tup` function performs a established computation on its argument, which has a value. On the other hand, the `apply_sec` function takes in an arrow type argument, which means this argument can be applied to its other argument.

3.2.4 Type Annotations and Inference

Rippl supports type inference through an implementation of the Hindley-Milner type inference system over the core syntactic language constructs of lambda abstractions, let-bindings, application, and variable names, as well as the additional language construct of if-then-else expressions.

By performing type inference in an environment where literals and language native operators (see section 3.5) are bootstrapped with a particular type, types of expressions can be inferred and checked in a complete and decidable manner. These type signatures may be concrete types like the ones below.

```
sum_ints :: int -> int -> int

count_chars :: [char] -> int

is_mutually_prime :: int -> int -> bool
```

However, these inferred types may also be parametrically polymorphic. These polymorphic types are represented as type variables like the ones below.

```
identity :: a -> a

len :: [a] -> int

empty_list :: [a]

list_map :: a -> b -> [a] -> [b]
```

Rippl also allows programmers to provide their own type annotations for function definitions. This is done using the Haskell-like syntax shown in the previous examples, which involves specifying the variable name followed by its full curried type signature, separated by a double colon "`::`". Type annotations are optional, but when they are provided, type-checking is performed to make sure the user-annotated types are a subtype of the inferred type.

The following function definition is an example of how an inferred type and annotated type can type-check directly if they are equal.

```
succ :: int -> int
succ n = n + 1       # inferred type is int -> int
```

The following type annotation yields a typing error, since the inferred type and annotated type contradict each other. This is due to the built-in type of the integer addition and division operators and the fact that Rippl doesn't perform any implicit type promotions.

```
avg :: int -> int -> float
avg x y = (x + y) / 2   # inferred type is int -> int -> int
```

The following function definition and annotation type-check correctly because the inferred type, which is parametrically polymorphic with type variable `a`, can be properly concretized by substituting `a` with the concrete type `[int]` to match the user-annotated type.

```
nest_int_list :: [int] -> [[int]]
nest_int_list l = l cons []     # inferred type is a -> [a]
```

Type annotations are optional for all function declarations besides `main`.

3.3 Grammar and Syntax

3.3.1 Semantics

The following grammar represents a high-level overview of Rippl semantics.

μ ::=		*Entrypoint*
	`main` $= e$	main method
e ::=		*Expressions*
	c	literals
	x	variables
	$e \sim e$	application
	`fun` x `->` e	lambda abstraction
	`let` $x = e$ `in` e	let binding
	`if` e `then` e `else` e	if-then-else
	γ	list comprehension
	`just` e	just constructor
	`none`	none constructor
	(e, e)	tuple
γ ::=		*List Comprehensions*
	`[`e`...`e`]`	ranged list
	`[`e `|` $(x$ `over` $\gamma)+$`]`	parametric list
	`[`e `|` $(x$ `over` $\gamma)+$`,`$e+$`]`	qualified list
σ ::=		*Primitive Types*
	`int`	integer
	`float`	floating point number
	`bool`	boolean
	`char`	character
δ ::=		*Higher-Order Type Constructors*
	`[]`	list
	`()`	tuple
	`maybe`	maybe
τ ::=		*Types*
	σ	proper type
	$\delta\ \tau$	higher-order type
	τ`->`τ	arrow type
θ ::=		*Type Annotations*
	x `::` τ	type annotations

3.3.2 Comments

Comments in Rippl draw from Python and Haskell. Single line comments are marked by a pound sign, while multiline comments are delimited by curly braces and dashes. Both forms of commenting support nested comments.

```
{- The following code will first declare a variable with value 2
   After that, it will be raised to the power of 5 -}

let x = 2 in
2 ^ 5 # 32... what a #cool operation!

{- All programmers eventually fall victim to writing
silly  {- superfluous -} comments, so it is important to
learn how to write {- concise -} documentation! -}
```

3.3.3 Keywords

There are several keywords that are reserved in Rippl and thus not available for use as identifiers.

```
{- Data Types -}
# int, float, bool, char, maybe, just, none

{- Boolean Logic -}
# and, or, not, true, false

{- Program Structure -}
# over, let, in, fun, if, then, else, main

{- Operators -}
# head, tail, cons, cat, len
# ,first, sec
# ,is_none, from_just
# ,int_to_float
```

3.3.4 Identifiers

Identifiers can be any sequence of characters that start with either a letter or an underscore followed by any combination of letters, numbers, and underscores.

```
my_identifier123              # Valid!
_my_other_indentifier456      # Also valid!
1_crazy_name                  # What the heck? NO!
```

3.3.5 Operators

The following subsections detail the various operators associated with each type in Rippl. It is worth noting that arithmetic expressions are evaluated using a mathematical order of precedence with left associativity. That is, expressions are evaluated using the following rules in order of decreasing precedence: parentheses, exponents, unary operators, multiplication/division/modulus, addition/subtraction. For the power operator, if the base of the exponent is a finite value less than 0, and the power is a finite non-integer, a domain error occurs and behavior is undefined as it is in C.

3.3.5.1 Integer Operators

Operator	*Type*	*Function*
+	$int \rightarrow int \rightarrow int$	addition
-	$int \rightarrow int \rightarrow int$	subtraction
*	$int \rightarrow int \rightarrow int$	multiplication
/	$int \rightarrow int \rightarrow int$	division
%	$int \rightarrow int \rightarrow int$	modulus
^	$int \rightarrow int \rightarrow int$	power
-	$int \rightarrow int$	negation
>	$int \rightarrow int \rightarrow bool$	greater than
>=	$int \rightarrow int \rightarrow bool$	greater than or equal
<	$int \rightarrow int \rightarrow bool$	less than
<=	$int \rightarrow int \rightarrow bool$	less than or equal
==	$int \rightarrow int \rightarrow bool$	equal
!=	$int \rightarrow int \rightarrow bool$	not equal

3.3.5.2 Floating-point Operators

Note that float operators differ from integer operators in that they require an additional '.' to aid in type inference.

Operator	*Type*	*Function*
+.	$float \rightarrow float \rightarrow float$	addition
-.	$float \rightarrow float \rightarrow float$	subtraction
*.	$float \rightarrow float \rightarrow float$	multiplication
/.	$float \rightarrow float \rightarrow float$	division
^.	$float \rightarrow float \rightarrow float$	power
-.	$float \rightarrow float$	negation
>.	$float \rightarrow float \rightarrow bool$	greater than
>=.	$float \rightarrow float \rightarrow bool$	greater than or equal
<.	$float \rightarrow float \rightarrow bool$	less than
<=.	$float \rightarrow float \rightarrow bool$	less than or equal
==.	$float \rightarrow float \rightarrow bool$	equal
!=.	$float \rightarrow float \rightarrow bool$	not equal

3.3.5.3 Boolean Operators

Operator	*Type*	*Function*
`and`	$bool \rightarrow bool \rightarrow bool$	and
`or`	$bool \rightarrow bool \rightarrow bool$	or
`not`	$bool \rightarrow bool$	negation

3.3.5.4 List Operators

Operator	*Type*	*Function*
`cons`	$a \rightarrow [a] \rightarrow [a]$	construct
`head`	$[a] \rightarrow a$	head
`tail`	$[a] \rightarrow [a]$	tail
`len`	$[a] \rightarrow int$	length
`cat`	$[a] \rightarrow [a] \rightarrow [a]$	concatenate

3.3.5.5 Tuple Operators

Operator	*Type*	*Function*
`first`	$(a, b) \rightarrow a$	first element of tuple
`sec`	$(a, b) \rightarrow b$	second element of tuple

3.3.5.6 Maybe Operators

Operator	*Type*	*Function*
`is_nothing`	$maybe\ a \rightarrow bool$	return `true` if none
`from_just`	$maybe\ a \rightarrow a$	extract element

3.3.6 Let Bindings

Given the functional nature of Rippl, there are no imperative assignment statements. Instead, Rippl consists mainly of nested expressions strung together using the let-in construct. This takes the form of `let identifier = expr1 in expr2`, where `identifier` serves as a name binded to the value of `expr1` to be used in `expr2`. The value of a let binding chain is the value of the last expression in the chain.

```
let x = 1 in
let y = 2 in
let y = 3 in
x + y + z # 6
```

3.3.7 If-then-else Expressions

Like most programming languages, Rippl supports basic control flow through if-then-else expressions in the following form: `if bool then expr else expr`.

```
let num = 5 in
let my_string = if num == 5
    then "five"
    else "not five" # "five"
```

3.3.8 Lambda Abstractions

One cornerstone of Rippl is to permit computations that require higher-order functions, which is made possible by lambda abstractions. These anonymous functions are written using the `fun` keyword and are arrow type expressions.

```
fun x -> x + 1 # int -> int
(fun x -> x + 1) 9 # 10 - application with an anonymous function
```

We can also bind these lambda abstractions to identifiers using a `let` expression.

```
let add_one = fun x -> x + 1 in
add_one~9 # 10
```

Hope you have a sweet tooth! Rippl provides some syntactic sugar when it comes to name-binding functions. The following expressions (after de-sugaring the syntax) are equivalent.

```
let add_one = fun x -> x + 1 # add_one :: int -> int
let cooler_add_one x = x + 1 # cooler_add_one :: int -> int
```

To top it all off, Rippl embraces the Haskell paradigm that all functions are curried. In other words, all functions in Rippl actually take just one parameter, either in arrow type form or proper/higher-order type form. This allows a Rippl user to employ partial application on functions!

```
let sum_three x y z = x + y + z in
# sum_three :: int -> int -> int -> int
let sum_two_add_one = sum_three~1 in
#sum_two_add_one :: int -> int -> int
```

```
let add_four = sum_two_add_one~3 in
# add_four :: int -> int
add_four 5  # 9
```

3.3.9 List Comprehensions

List comprehensions are an elegant and concise way to define and construct a list. They use a well-known mathematical notation to substitute complex list operations like map and filter.

3.3.9.1 Ranged List

With ranged lists, we can create a list without specifying every element. Ranges will always have an interval step of +1 and are only allowed for integer lists.

```
[18...24] # [18, 19, 20, 21, 22, 23, 24]

['a'...'z'] # not allowed

[1.2...2.7] # no sir, uncountable number of elements
```

3.3.9.2 Parametric list

Parametric lists allow users to specify a parameter.

```
[x * 2 | x over [3, 1, 4]] # [6, 2, 8]

[x | x over [9...12]] # [9, 10, 11, 12]
```

3.3.9.3 Qualified List

With qualified lists, lists can be filtered by one or more conditions that are represented as computations on the bound list parameters and return a boolean value.

```
[x ^ 2 | x over [1...10], x % 2 == 0] # [4, 16, 36, 64, 100]
```

3.3.9.4 Parallel List

Parallel list comprehensions are lists comprehensions with multiple parameters.

```
[x - y - z | x over [10, 1], y over [2, 3], z over [1, 5]]
    # [7, 3, 6, 2, -2, -6, -3, -7]

[x + y | x over [10, 30, 50], y over [10...12], x != y]
    # [20, 21, 22, 40, 41, 42, 60, 61, 62]
```

3.3.9.5 Lazy Evaluation

Rippl uses lazy evaluation for list comprehensions. This means that only the head of the list is initially fully evaluated and all other elements are evaluated when their value is requested. We can see the consequences of lazy evaluation in the example below:

```
let funky_list = [1/(x-1) | x over [0...1]] # no error
    # even though when x == 1, 1/(x-1) is undefined
head funky_list # -1
    # only evaluates the first element of the list
tail funky_list # causes an error
    # will evaluate the second element, which divides by 0
```

We can see that when only the first element of the list is evaluated, there is no error. Although the second element of `funky_list` will have an error from dividing by 0, this element is not evaluated until we get the tail of the list. When we do get the tail and thus evaluate the second element of the list, there will be an error.

3.4 Immutability

As a pure functional language, Rippl enforces immutable semantics. This means that once an expression has been assigned to a variable name, the value can't be changed later on in the program (inducing a change in state). Rather than performing actions on objects and altering them, Rippl performs computations that return new values. This protects programs from side effects that may occur from having having multiple operations computing and mutating the same data.

Take for example the code snippet below. The use of the `cons` operator returns a new list with a first element of `0` rather than changing the value of `natural_nums`.

```
num_set :: bool
num_set = let natural_nums = [1...1000] in
let whole_nums = 0 cons natural_nums in
(head natural_nums) == (head whole_nums)    # false
```

By nature of being a functional language, Rippl has no constructs supporting reassignment of variables. However, in the case of nested let bindings, the program semantics may resemble mutability in variable reassignment. The program shown below for a definition of `change_a` is an example of such a case. Nevertheless, it's important to note that this is not a case of mutability, but a case of variable shadowing. The `a` in the final expression binds more tightly to the `a` assigned in inner let binding so the final value returned is that of the second `a`.

```
change_a :: int
change_a =  let a = 1 in
            let a = 2 in
            a       # 2
```

For a more concrete example of how this semantically differs from mutability, consider the modified definition of `change_a` below. By the second let-binding, the local definition of `a` binds more tightly than the higher-level binding for `a` and instead becomes a recursive definition. This results in a program error in both Rippl and Haskell.

```
change_a2 :: int
change_a2 = let a = 1 in
            let a = a + 1 in
            a       # error!
```

3.5 Entrypoint

Like all pure functional languages, Rippl ensures that the evaluation of all expressions is free of side effects, or stateful interactions with the outside world. This prohibits the insertion of print statements inside functions that execute as they are evaluated.

Also as a pure functional language, Rippl guarantees that any computation given an argument will always return the same output. This means there can be no assumption of order of evaluation in a program as there is in the imperative paradigm. This prohibits the use of language constructs like `input` or `scanf` that read and return a value read from the user with no arguments (save the format-friendly strings present; this is more like a separate print/IO action than a true argument). With no varying arguments, the function call in a pure language should return the same value each time which should not be the behavior of an IO operation.

Pure functional languages like Haskell get around this by giving all the IO function calls a hidden argument and strings each call together as arguments and dependencies to create a proper ordering by using an IO monad. This operation is sugared up into an imperative-looking `do` construct in their main methods.

Like in Haskell, the main method in Rippl is the top-level entrypoint into a program. In order to avoid the use of higher-kinded types like in Haskell, the main method is used such that there is only one output operation allowed. There is no reading of user input.

The evaluated value of the body of the main function is printed. The ”Hello, world!” program is written as follows in Rippl.

```
# this prints "Hello, world!" to the terminal
main = "Hello, world!"
```

If the right-hand side of the main method were a more complex computation expressed in terms of other function definitions provided in the file, then the full expression would be evaluated and printed.

It might be noted that this construct of IO in a language doesn't allow for a function not to have output (Rippl doesn't have a `void` value representing a bottom type). However, this design should be reasonable since Rippl currently supports no other output operations so programs would be moot otherwise.

4 Project Plan

4.1 Planning, Specifications, and Development

We primarily worked on a VM instance running Linux Ubuntu 18.04 on a Google server – CLAC. This allowed us for standardize our development with consistent versions of LLVM, OCaml, zsh, and gcc. As for planning, we agreed upon regularly set meetings (every Friday morning) where we first went over our accomplishments for the week and plan out what the next week would look like. All specifications were agreed upon at our meetings where we also enforced these standards.

4.2 Project Timeline

January 30th, 2019: Rippl concept born
February 13th, 2019: Rippl proposal submitted
February 19th, 2019: Work on scanner begins
March 4th, 2019: Work on parser begins
March 11th, 2019: Scanner and parser pretty much completed
March 18th, 2019: Semantic transformation: type inference and lambda lifting
April 1st, 2019: Rudimentary code generation for "Hello World" completed
April 15th, 2019: Lambda lifting completed
April 17th, 2019: Backend representation and C library work begins
April 19th, 2019: Type inference completed
April 22th, 2019: Code generation overhaul begins
May 15th, 2019: Code generation completed
May 16th, 2019: Presentation!

4.3 Roles

Hans Montero (Team Manager):
Planning, work delegation, general time-keeping
Amanda Liu (Language Guru):
Language designer, backend representation creator
Hollis Lehv (System Architect):
Compiler designer, environment set up
Da Hua Chen (Tester): Test plan, test suites

4.4 Software Development Environment

VM: Linux Ubuntu 18.04 LTS (CLAC)
Text editors: Vim, Sublime
Programs: gcc 7.4.0 or clang 6.0.0, llvm 6.0.0, ocaml 4.05.0, zsh 5.4.2, opam 1.2.2, ocamlbuild 0.12.0

4.5 Project Log

```
commit 0f08acd54d31e5b78fa5fa0202ff73338168f5e5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Jan 22 22:56:47 2019 -0500

    Start rudimentary scanner with syntactic language basics

commit a9f65e9b79115ab290f7c06eea3ab51853cb162d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Jan 23 11:02:51 2019 -0500

    Move to src folder

commit 59fa4a1a7c6d37fad3db85d2f17fad4f56de2f34
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Jan 23 11:16:28 2019 -0500

    Test Travis CI

commit a9586cd2d360fcd13df6330771375a36131dff35
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Jan 27 18:18:46 2019 -0500

    Add test directory with examples

commit d374e9c1cfd44c1e9aa0136bbad081a4ffda0088
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Jan 27 18:29:11 2019 -0500

    Add fold implementation

commit 878995b4db1226b8b45b78be91978e9bbdf258ac
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 1 14:43:46 2019 -0500

    Add num sensitive operators

commit 36077172eaf4b53568380d8e329f9766bdd8542c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 1 14:44:04 2019 -0500

    Add more example programs

commit 867a14ed6757d7b858cb41404a54c222c34df11d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 14:53:27 2019 -0500

    Rewrite examples with Rippl syntax

commit 96c32a3821cece15b80c48f1ef58b6e0ebb23f9e
```

```
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 14:53:59 2019 -0500

    Remove msk examples

commit eea172b12474657ce732e71d1d0bbd319a451d9f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 15:01:06 2019 -0500

    Add mandelbrot program

commit 5ff0ec7fc68c56eac9aa8488acf5119dd44558a7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 15:34:00 2019 -0500

    Add new Rippl syntax

commit 620b075feaadce7ae61e655997ea6c6b19e05eb2
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 15:35:08 2019 -0500

    Add carrot power operator

commit 1c255cdda478d90408480a529d524b10656d8918
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 15:36:48 2019 -0500

    Fix shell test.sh XD

commit 028652d9f1cbba3e08f471efb563b1a5030cf287
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 22:00:40 2019 -0500

    Change float to . notation

commit 5cb7911d3303bd2bbb3208486e67567810e1e66e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 22:21:03 2019 -0500

    Change length to len and other syntax changes

commit a2350a3ae0929972a8e40f55aef953738ecd3f4f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 8 23:22:59 2019 -0500

    Add Collatz Conjecture

commit e3c2309823431747eb01b282c55c3cb2660e91c2
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Feb 10 20:31:08 2019 -0500
```

```
    Add proposal tex source

commit 0dc865a9834953ff7f6f08d6c16d0d078d9d5581
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Feb 10 20:32:46 2019 -0500

    Remove integer truncation note

commit 53c479c3292f8c7ebe8bbbe5ec4717f52a9dd7ac
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Feb 10 20:37:55 2019 -0500

    Change ++ to cat

commit 997990f8d74f0a95f199056cb7a5beeda5ce6b6e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Feb 10 22:25:24 2019 -0500

    Add infinite list example

commit d4ad003600e7918937cd3ad0765bca30760c2b6e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Feb 10 22:35:55 2019 -0500

    Remove Mersenne example

commit 6203973469c1fa4448ec179cd5c5db1ae1720ff8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Feb 11 15:13:44 2019 -0500

    Match for identifiers

commit cc026f930c1e0e91c5d2cfc7b823479f4bcdd42b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Feb 11 16:25:47 2019 -0500

    Pre changed to cons

commit 4773fcafd834c53f0448ccedbb14e26a58b29cae
Author: hlehv <hlehv@aol.com>
Date:   Mon Feb 11 19:51:30 2019 -0500

    Added introduction to proposal

commit 994d20ba8a7e028838f335f5bbff261d934c4829
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Feb 11 21:49:25 2019 -0500

    Change pre to cons in tests

commit 2b79f5774533b235fda44dfd48d4c379842b903d
```

```
Author: Hans <hjm2133@columbia.edu>
Date:   Mon Feb 11 22:17:27 2019 -0500

    Add language features

commit 9e0c32e1e623258cb8ccbae1d9903bbc9a68e94f
Merge: 2b79f57 994d20b
Author: Hans <hjm2133@columbia.edu>
Date:   Mon Feb 11 22:18:17 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 653db9c276da1b48edfc4c1789ccaea277121f38
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Feb 11 22:30:08 2019 -0500

    Fix subsection escape typo

commit e20500903efe306ae34ed3a34dbf2b5b79484b83
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Feb 12 09:53:01 2019 -0500

    Added draft of Motivation section

commit 4cb35278862d68e4b5d02dfa48ca18d1ef88d373
Author: Hans <hjm2133@columbia.edu>
Date:   Tue Feb 12 13:53:36 2019 -0500

    Revise language overview

commit 89e535a8099f554494f77bf7b7a99acff68926b1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Feb 12 14:06:51 2019 -0500

    Add another sentence clarifying the numeric type operators

commit 353ec3fc5e07e12eb5f63465aca3f9e666b5ec84
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Feb 12 19:04:17 2019 -0500

    Add new motivation/intro and new type inference sections

commit 19dbd737bc184351a5c2b186fcf277408d86f119
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Feb 13 13:17:58 2019 -0500

    Change dot to arrow in lambda abstraction

commit ab97e3316b6dbcce46485057a32bb201786a1549
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Feb 13 13:24:59 2019 -0500
```

```
    Change arrow to monospace

commit 9839b3ea82c4ebd29911b6d1afe3280fd046d78e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Feb 13 14:27:24 2019 -0500

    Fix margin size

commit e3c3df1bf19b2e0e650df43045a66f6fb17558eb
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Feb 13 16:51:55 2019 -0500

    Fix type inference and lazy eval

commit d25a12622a46b2e7c825e7ea2f258d3e0d0781f6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Feb 13 18:41:08 2019 -0500

    Fix margins so features is on one page

commit fe2001b2c247ba55c0018fc5f6095e61a76c5ea0
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Feb 13 22:20:45 2019 -0500

    Correct minor mistakes

commit f54e39ae65e6b68a2820831d39eb7a274e9320f3
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Feb 13 22:22:34 2019 -0500

    Compile Latex using Overleaf because VSC sucks

commit 0cec1a85724ce6133e918e8eff10d1de3a3daa0d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Feb 13 22:30:18 2019 -0500

    Hyphenate words

commit 35645a7d4a11b2bca24e94bc6d293c578fb14a1b
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Feb 13 22:31:48 2019 -0500

    Hyphenate another word

commit e971f937fd6a9cec850f8cdc29013e4036e53e6d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Feb 13 22:33:57 2019 -0500

    Trash old proposal
```

```
commit 8e6d53ebe60db074358a37d13b3e11207e0d7db9
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Feb 13 22:36:29 2019 -0500

    Add detail to list and numeric types

commit 5ea179c64ff95a51c602addd86615c0480d55d45
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Feb 13 22:39:09 2019 -0500

    Fix list spacing

commit 541cc2d8b265900e013f31ca5c690fe79b88c10c
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed Feb 13 22:41:33 2019 -0500

    hyphen

commit 0e892b15b626bf27eeb13277073a4ff002449e07
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Feb 13 23:07:14 2019 -0500

    Upload proposal

commit eeac3c83583a5cdbfa8e3d1e55cac0f1d6d1788f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Feb 13 23:16:18 2019 -0500

    Delete because I am paranoid

commit 2d00743b8b1a00f4696fc1a4c8441f6db81c25dc
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Feb 13 23:16:35 2019 -0500

    Upload the super final proposal

commit 24dcec0a4c200219b483145bf64eb6dd94905b6c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 10:31:42 2019 -0500

    Add comments to scanner, must add strings

commit b9a807f979632ca2e601ce0b7cb014a01018adfe
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 10:46:09 2019 -0500

    Start rule for chars

commit 06d69959706da228f844836e850e21d6b7564518
Merge: b9a807f 2d00743
Author: Amanda Liu <al3623@columbia.edu>
```

```
Date:   Fri Feb 22 10:46:32 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit bb2b85c8e805ef10d08dd54d26c90037e651a434
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 10:52:32 2019 -0500

    Add char rule I think

commit 48fbbdb608b52c76912d623fca4a32810876e421
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 10:56:16 2019 -0500

    Add single line comment

commit 9a64e8f7a16ecde0e228445f4d3dc9f25e6cdb9d
Author: Hans <hjm2133@columbia.edu>
Date:   Fri Feb 22 12:21:11 2019 -0500

    Allow underscores as first letter of indentifiers

commit ca87f12ce0ac51268e90d2615baff06efda09692
Author: Hans <hjm2133@columbia.edu>
Date:   Fri Feb 22 12:24:54 2019 -0500

    Consider carriage return in white space

commit ba0341b2749d0b598392189fb4d7d94b3ab47a80
Author: Hans <hjm2133@columbia.edu>
Date:   Fri Feb 22 12:39:07 2019 -0500

    Enforce consistent spacing and naming

commit e4062d773f379f592a8824100ca9bf6e67bfe8cf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 13:10:10 2019 -0500

    Lex escaped slash

commit 384d94fcefe98445d2452e17605914ca7dc7ba0a
Author: Hans <hjm2133@columbia.edu>
Date:   Fri Feb 22 13:19:04 2019 -0500

    Change comment style

commit dab7b9f6c551d2624437c599a056c54d136e4aaf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 13:19:44 2019 -0500

    Add newline and tab char
```

```
commit 3fe6e2731a02896cc4029c16c8a7c1b8967baf32
Merge: dab7b9f 384d94f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 13:19:50 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit f8aca9327a8675a387e2f1b49b8669d31d565aaf
Author: Hans <hjm2133@columbia.edu>
Date:   Fri Feb 22 13:20:50 2019 -0500

    Add one space

commit bbd92df7324b3831c0e9285af6ba578e76d341db
Author: Hans <hjm2133@columbia.edu>
Date:   Fri Feb 22 13:22:53 2019 -0500

    Add carriage return literal

commit 09ac715d6bb5e1a3408bff78c551eb9115391d13
Author: hlehv <hlehv@aol.com>
Date:   Fri Feb 22 13:35:38 2019 -0500

    added return charcter

commit 9f8a67287e40ccf80683786e4348fcf4df965c90
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 17:37:32 2019 -0500

    Add call back to lexing after end char

commit 8b838c2426c84f6844e5b2f78a470ed84371084c
Merge: 9f8a672 09ac715
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Feb 22 17:37:46 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 41b24d823169147f884bf739a5acc880d9ab3d35
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Feb 25 15:02:40 2019 -0500

    Fix typo

commit baa02bdacad48ec2bdf6615bc6fba4745fad1a6c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Feb 25 15:06:32 2019 -0500

    Fix end char_char_literal
```

```
commit 74ecf8ae3aa3f74136877c2aee2be6515d86be8f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Feb 25 15:16:19 2019 -0500

    Add float lexing

commit 4b0ce60d3aef8e563121bd9bc9085968c7006c90
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Feb 25 15:19:26 2019 -0500

    Allow lexer to take negative int literals

commit 123d106ac4d1ce2f9713910ffc53823a22b5eb6f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Feb 26 19:01:59 2019 -0500

    Restructure test suite

commit ee9b42ae3beda3a7534459658c83595dcadb1ebd
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Feb 26 19:33:51 2019 -0500

    Add lexer tester to scanner.mll

commit 57f2c333c5e261a6c1e3205f1032ca1be129bdd0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Feb 26 19:37:58 2019 -0500

    Add temporary parser

commit 6cf3abeeb9da3727f03eb73df02428108066a712
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Feb 26 20:22:49 2019 -0500

    Simpler chars and strings

commit 24439d7a30deacd5a2bbf168468562dafc677fc7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Feb 26 20:34:04 2019 -0500

    Make print literals better

commit 9b7b05296fc0695a359e21d4441e91842e20f2af
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Feb 27 16:17:28 2019 -0500

    Remove old driver

commit 638d49440903bae1de83a7db05ca0a2fd591ea83
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu Feb 28 18:56:43 2019 -0500
```

```
    testing of lexer

commit 4e5bbe2b311172c1954dddd132dc5a3eb5073752
Merge: 638d494 9b7b052
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu Feb 28 18:57:28 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit f20d1b1a18ae8bda1bef8e5f35aa63600eabca62
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu Feb 28 19:07:14 2019 -0500

    LISTCOMP -> BAR name change

commit 22edaabc513a15720b6c5785470f7023fdcfa7d9
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu Feb 28 19:49:16 2019 -0500

    updated lex test script

commit d3b08ea270ba848330f960ca0de591beb0861e9c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Feb 28 20:27:35 2019 -0500

    Add possible solution for parsing LRANGE

commit 5539e2bab36d993acf420ba57eb0b63688b023f2
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Feb 28 20:32:04 2019 -0500

    Update parser and make lexer file symlink

commit 59576e64beace5149f6b5ed46cf4247ddf339971
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu Feb 28 20:46:53 2019 -0500

    LRANGE defn reverted in test driver

commit 1376fdc72e871d21e49c1230bf729f344f748c0a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Feb 28 20:58:16 2019 -0500

    Fix float lexing ambiguity

commit e719843bc9cbc670d11da6528162414ff9e7dd71
Merge: 1376fdc 59576e6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Feb 28 20:58:30 2019 -0500
```

```
    Merge branch 'master' of https://github.com/al3623/rippl

commit 06d64541c1bcaa457b869526615cf87815810138
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Feb 28 21:03:30 2019 -0500

    Add bar

commit fa48330954f1dfa5453375379138acb158ca6ef2
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Feb 28 21:39:18 2019 -0500

    Make all tests work by adding types and fixing ellipses in range

commit 74fbc621da2c3a17d0990ecfaf384febbfef133f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 10:36:07 2019 -0500

    Add or to io proposal

commit 4b564e5f2cde753e2ee419de66f64d0af46d7fb8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 10:56:29 2019 -0500

    Add lexer tests and test lexer

commit 76484204102b09898e52084fa757d31239b56b80
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 1 11:01:13 2019 -0500

    Add a complicated test... to do: fix multiline strings

commit ca61bec3fa777ff8abac8f824b7024ffe7e9181d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 11:30:18 2019 -0500

    Finalize io example

commit 8d6c56f8e3e5e693b6b08954ef74fb7cd89bd9d1
Merge: ca61bec 7648420
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 11:30:31 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit cbd7f3c02f7e56b40f84b67efbd2748f777000ea
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 1 13:06:20 2019 -0500

    Create README (lowkey, TODO)
```

```
commit a25dec9d774481cef6228bd20568fa886b4ab9a8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 13:14:09 2019 -0500

    Copy proposal for baseline LRM tex file

commit 71dba2c895094aa3aabbedc2fa799198cc67c859
Merge: a25dec9 cbd7f3c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 13:14:22 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit e20026261c73319b4ec9f04ffcb2955e4a3e75aa
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 1 13:15:24 2019 -0500

    Add main keyword

commit cdbb585f62f3609e4bdabe67580378628e245a55
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 1 13:16:18 2019 -0500

    Update TODO

commit bdfb8e7bdd3cc6a2f04fab3f47d2cb70a8253e90
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 13:30:01 2019 -0500

    Add main method to syntax and change titles

commit 2c8bc1264ac3faba68b782bab9fa648ac7f63b80
Merge: bdfb8e7 cdbb585
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 13:30:20 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 04149e64743f19729e7768ba38daa98edf018142
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 1 13:40:01 2019 -0500

    Changed Hollis's title in LRM.

commit 63964487c905b5409f017800c53fdf7c7fc26fc9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 14:03:52 2019 -0500

    Add type annotations to grammar and restructure
    proper/higher order types
```

```
commit 26eecfc5e7227c3f9461777be6a7bfd9c7f17392
Merge: 6396448 04149e6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 14:04:07 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 92902e996175b6d98fa8c887cebe369cb084107b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 14:04:39 2019 -0500

    Update IO

commit df67c262be21bbe52cd74e8636de807852d2dc9a
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Fri Mar 1 14:05:18 2019 -0500

    Cross out main and type annotations from TODO

commit ab81018e84d19876a2985e52961a708d4925d631
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 16:32:23 2019 -0500

    Start AST

commit a95f6953701fc960aff6a47b8a05b8ac3ff5fc8a
Merge: ab81018 df67c26
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 16:32:37 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit b7133ff45fa630d32ea1a1f019826481221da259
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 16:38:45 2019 -0500

    Add type heirarchy

commit 3a689be36abc1f55a5c6c5438edf58b8e9ddcb3d
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 1 17:05:27 2019 -0500

    Added list comprehension introduction and code examples.

commit 5051076e6e3d908a2b1195b50ed5764c198e3192
Merge: 3a689be b7133ff
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 1 17:05:31 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl
```

```
commit 184d1e2c5253cf3cfc0c7b858bbeb2a1a2e6da55
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 1 17:26:47 2019 -0500

    Added qualified list, parametric list, and ranged list
    definitions and examples

commit 8eb48efc6989dcb9064feee40bfc1f4e1fcf9ca7
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 1 17:29:06 2019 -0500

    Changed ripple to rippl

commit 489d241beaae072989477ac46cf68cea56c8a981
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 1 17:43:41 2019 -0500

    Added parallel and infinite lists

commit 860b45049df3f578af158c54da211211b7b2e9af
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 1 17:54:05 2019 -0500

    Changed grammer and syntax of list section and changed some list
    comprehension output

commit f28362a72626ba6c800da0851e8a85cab9027da5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 1 23:58:58 2019 -0500

    Add comment syntax highlighting for LRM

commit 8a8dfdfae82bd74738a7d9d738525341d0868d00
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Mar 2 00:09:34 2019 -0500

    Add IO code sample and make list comp. its own section

commit 22ab920003eca655b5b1414e1c870d61b1e5fdae
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Mar 2 00:14:01 2019 -0500

    Add list comprehensions to formal grammar/syntax section

commit 939be18b4b0b7a4b11d3f3ad61ba0f7977e86cad
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Mar 2 00:19:03 2019 -0500

    Change comment documentation to match hash and brace-dash

commit 4daf076368319c62a479271010f5f8df75e76839
```

```
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Sun Mar 3 18:05:22 2019 -0500

    Cross out code samples TODO

commit 8cd5357ab25e16caa91e94f1a0c5161fe621c489
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 3 19:00:58 2019 -0500

    Add IO/main section to LRM and restructure

commit 7a15fe044a7c9fe228c76451753c0627a09f8126
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Sun Mar 3 19:01:30 2019 -0500

    Cross out main/IO TODO for LRM

commit 94441b0d2e19dcdc1967100c229847413f913819
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 3 19:04:45 2019 -0500

    Add section for Type System

commit c60d035599c606c511f925db58ac1c7a748e1da8
Author: hlehv <hlehv@aol.com>
Date:   Sun Mar 3 19:05:42 2019 -0500

    Updated todo's with list comprehensions completed

commit 4c96f4e757e912f880265419fe6af5245409db53
Merge: c60d035 94441b0
Author: hlehv <hlehv@aol.com>
Date:   Sun Mar 3 19:05:48 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 99e52392133a4bd151c57efe61a5fd257cc552f0
Author: hlehv <hlehv@aol.com>
Date:   Sun Mar 3 19:11:24 2019 -0500

    Fixed TODO formatting by removing crossing out from bullet point itself

commit 7394908412733db5c5a1b95583f464445fa3b1e5
Author: hlehv <hlehv@aol.com>
Date:   Sun Mar 3 19:12:35 2019 -0500

    Really fixed TODO formatting this time, cross out all
    bullet points completed by Hollis

commit 452ee87b33594c81b4149797f7f9c61572e651ef
Author: Amanda Liu <al3623@columbia.edu>
```

```
Date:   Sun Mar 3 19:15:59 2019 -0500

    Add polymporhic type signature for main

commit 0e8159c72317fe30ab81d4ae4d1da764241f7897
Merge: 452ee87 7394908
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 3 19:16:12 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 7f598a275efae5e5c42ef4229e24e62d208c0783
Author: hlehv <hlehv@aol.com>
Date:   Sun Mar 3 19:21:55 2019 -0500

    Added another test for the lexer for list comprehensions

commit c984f7931b2e8b1c29c0ac78b29f73bcb4f614a2
Merge: 7f598a2 0e8159c
Author: hlehv <hlehv@aol.com>
Date:   Sun Mar 3 19:22:01 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 7481dd9efb5b925a36580be39cf781d230a81e3a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 3 20:26:40 2019 -0500

    Add type annotation section, remove space underscore, add cons keyword

commit d376011e0c468817566b5cf604498ab15d955eae
Merge: 7481dd9 c984f79
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 3 20:26:57 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit e32c337f2fadabb432c509c5ea7e4bb0d0ebddfe
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Sun Mar 3 20:28:47 2019 -0500

    Cross out type annotations and inference from TODO

commit def96c248f96ea35db71321683906b5f77d71154
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 3 20:30:35 2019 -0500

    Fix typo (slashes in strings)

commit fbf914bb4b9e41b1fdc2c345f5f685b811898d8b
Merge: def96c2 e32c337
```

```
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 3 20:30:46 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 67835365d6692133348dac5019f144b2ba6e2701
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 3 20:39:28 2019 -0500

    Fix lang definition order so all keywords highlight

commit d9e41e054143bc851d8233aaecfe7949e40aa6f4
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Sun Mar 3 23:44:18 2019 -0500

    Added most of Type System section

commit 0efb8e38e8cbb259ccf1e77022f866fd6748dd30
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Sun Mar 3 23:56:18 2019 -0500

    package and newcommand for single quot

commit 74b56b466bc4efd602d2c75b024906b118881861
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 00:20:10 2019 -0500

    Add half of grammar/syntax section

commit 38c463f50a3ff516c3a175bf98c8e57ac3def208
Merge: 74b56b4 0efb8e3
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 00:20:13 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 52548456e4745b89717315af3e3aa9c4c6cac638
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 00:34:32 2019 -0500

    update todo

commit c21d0144463bf3759b9035ff01ef32039816ba9a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 00:35:08 2019 -0500

    Update todo again!

commit bc09f143c6d7fe5cbe19c3fbb6be17e07261cb1d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 03:23:25 2019 -0500
```

```
    Finish syntax, clean up other sections a bit

commit ee4742ae01d775062919e0a4bb2beabe11f084ce
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 03:23:41 2019 -0500

    Add table of contents, fix a typo

commit 7a410d15b76f65fa481b6fcfe85b988bed2068c1
Merge: ee4742a c21d014
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 03:23:54 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 00c11d15eae556906aaf7a2cc3e0edf0e89e72ff
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 03:25:42 2019 -0500

    Update todo

commit 15a68e4279d8a153470e74b12f78fb395a4e0d38
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 03:29:05 2019 -0500

    Reference another section

commit e7cde4d83e0f307e5d82af2a09a8337b45a5d5b8
Merge: 15a68e4 00c11d1
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 03:31:11 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 0745152d8e1a64ace4a92d661d57d453717c9006
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Mar 4 13:08:37 2019 -0500

    Update LRM.tex

    Adding multiline comment syntax highlighting, explicit
    subsubsections for proper types, changed operator subsection names,
    Altered Da's nickname

commit 7f016419776a0745130eb42d18908c465285e4bb
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Mar 4 13:40:11 2019 -0500

    Title change, added email addresses
```

```
commit 50640a2d358a5133d3377156917106cfd24ff3c7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Mar 4 14:40:11 2019 -0500

    Add possible title pages

commit 0875d0acdd9563055904ea922738c37c3be36e77
Merge: 50640a2 7f01641
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Mar 4 14:43:16 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit aee5a26eba5e8fcc3e9633e49f6b1c6024975a1f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Mar 4 16:09:41 2019 -0500

    Get rid of two sentences

commit b4e4b76be1a2d084470eb5288c0cad3aa4f1f2e9
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 4 17:27:20 2019 -0500

    Proof read just a bit

commit b6e5ce8b77c43eb6c169cf7610144a9e5111eb7a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Mar 4 17:35:57 2019 -0500

    Remove indents

commit 4740f67e4f066309f33950e3ff5a351c108daf9d
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Mon Mar 4 17:37:22 2019 -0500

    Update TODOs

commit b2e8b7b26d56709f54555b234465a131ebe9c016
Merge: b6e5ce8 4740f67
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Mar 4 17:38:31 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 671be55cb2f81c252d0b07884d0332d305989d84
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Mon Mar 4 17:42:32 2019 -0500

    Update README with assignments

commit 5e5b36346f786938fe2b58259dece0941645e177
```

```
Author: hlehv <hlehv@aol.com>
Date:   Mon Mar 4 23:15:43 2019 -0500

    added main to scanner.mll, lexdriver.mll and parser.mly

commit 9313d72e72450f720136f8c8a850e4cf046228de
Author: hlehv <hlehv@aol.com>
Date:   Mon Mar 4 23:32:21 2019 -0500

    Removed minus from integer and float literals

commit 788e5d32115e9f86f7d0c90c795ed2d87a7496e1
Author: hlehv <hlehv@aol.com>
Date:   Mon Mar 4 23:46:09 2019 -0500

    Removed weird entry from infinite list section

commit bdc1c2f4cabf1bab18928458b32943ad278ec4e4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 5 00:01:46 2019 -0500

    Add maybe type to formal syntax

commit 823ef125e9403e9dccd65e5a3587e4496262056c
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Mar 5 00:02:20 2019 -0500

    Cross out maybe type formal syntax from TODO

commit b830369cddd4c531aa485ca00044980e289c8b63
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue Mar 5 01:17:10 2019 -0500

    add maybe operators

commit 6ec24ccef1374aafa3556ff1ce738d289b88dc51
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue Mar 5 01:19:18 2019 -0500

    Update TODO for maybe

commit f076959b17bdf7e0f12bc585c8263690cd5825cd
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue Mar 5 09:51:39 2019 -0500

    Add Makefile for lexer

commit 88b136c9911a1b1dd67ec1c65789bb799fb89778
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue Mar 5 10:00:46 2019 -0500
```

```
    Remove epilogue from lexer, add Makefile that creates lexer driver

commit 0ec4772af4071d89e354a8c6322815fc2ba789b9
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue Mar 5 10:06:56 2019 -0500

    Add more action items for driver

commit b42e3716192d7b8fabe7bf834270904886e53768
Author: hlehv <hlehv@aol.com>
Date:   Tue Mar 5 14:58:54 2019 -0500

    Added example to list comprehensions about lazy evaluation

commit db9ea956c15631246f2504d6b3cce69a00fe049b
Merge: b42e371 0ec4772
Author: hlehv <hlehv@aol.com>
Date:   Tue Mar 5 15:03:36 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit b8117473e55734f2139541e04d0405003d5fd969
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 5 15:14:39 2019 -0500

    Restructure lexing tests with script to run all tests

commit 26a29367c140f67076588f6debaf0d3589f890bf
Merge: b811747 db9ea95
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 5 15:14:52 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 381c503a35d9319c8aa4535178340684ee57f87f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 5 15:22:59 2019 -0500

    Add minor edits to list comprehension section of LRM

commit 4ce55a104e983bb452db6e0c3e3565064b71999a
Author: hlehv <hlehv@aol.com>
Date:   Tue Mar 5 20:53:16 2019 -0500

    Reordered list comprehensions section

commit 152256d1ac888c68d4af97933f4f5e38559f6d75
Author: Hollis Lehv <33107997+hlehv@users.noreply.github.com>
Date:   Tue Mar 5 21:07:56 2019 -0500

    Update README.md
```

```
commit 1221412f5b16b7760763d86c3d7fc64f70314ec3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 5 21:47:24 2019 -0500

    Add logo jellyfish and png

commit c757b455bf97f5377768a8918344eccd84f93f9a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 6 11:16:39 2019 -0500

    Add immutability section to LRM

commit e082de021c2ba3b3118a8b2ccb844ae814c40351
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 6 13:08:21 2019 -0500

    Fix typos in immutability section of LRM

commit 0586d139a7a2c34407d56f50d88ca53a301ee3de
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 6 13:11:17 2019 -0500

    Remove extra pagebreak

commit 393734f53dd1ed1a762ca6415a42f3220bd4dd71
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed Mar 6 13:15:20 2019 -0500

    Add logos

commit a8916f830283c63417c5c51e455583c049843e57
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed Mar 6 19:59:24 2019 -0500

    Update LRM.tex

commit 097c61175efdb73c35803507d005c9ab2e121255
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed Mar 6 20:05:39 2019 -0500

    logo swap

commit ef2f1d1d34022385e5bfd21fd6a4713885db260f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 6 20:27:10 2019 -0500

    Add LRM to LRM

commit 260136dc913546234694fe209b8bcfbb3a575d01
Author: Amanda Liu <al3623@columbia.edu>
```

```
Date:   Wed Mar 6 21:01:42 2019 -0500

    Fix centering on titlepage

commit 00becf21ce1ce47e7988ae21130a5b5f132b69cf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 6 21:04:42 2019 -0500

    Fix line spacing

commit 5a77a8865207123112f8060068e0b3fd4ef0ae6bc
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 6 23:14:57 2019 -0500

    Fix indents

commit 8c78b98203c2b28aa0a1e8788aef2ade4113d461
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu Mar 7 00:05:57 2019 -0500

    Make eg's consistent

commit a87b2cf612a1785df52b9b920ce31ecfbd37a8bd
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu Mar 7 00:06:39 2019 -0500

    Upload LRM

commit 4b0cf153ec3a3453877b23058aef961cdcd59c8a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu Mar 7 00:07:18 2019 -0500

    Update TODO     LRM finished wahoo

commit 04d2d171b06fe2265e55af81f1b5447d1c562a1c
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu Mar 7 21:03:56 2019 -0500

    Add MAYBE, JUST, NONE, LEN to parser, lexer, and epilogue

commit 99891668a79dc1871fcd58d520112150f62b2dee
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu Mar 7 21:04:21 2019 -0500

    Update README.md

commit fc46907b9247878261bf723dae4d9803f53bae61
Author: Hollis Lehv <33107997+hlehv@users.noreply.github.com>
Date:   Fri Mar 8 11:45:21 2019 -0500

    Update README.md
```

```
commit 0b5ee38d49856d3715a843e680d5cd40558fb7e4
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 11:57:45 2019 -0500

    Changed unary minus in scanner.mll and lexdriver.mll,
    added to parser.mly

commit dccdc1183d65cef19a4bf983bf3a29e9a8e76e46
Merge: 0b5ee38 fc46907
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 11:57:51 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit 15732906c09ed353006e54ef51c9d8e80b9a71f6
Author: Hollis Lehv <33107997+hlehv@users.noreply.github.com>
Date:   Fri Mar 8 12:03:13 2019 -0500

    Update TODOs

commit 14dbb41bed9c8e83108583228fab240fa5da2e56
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 12:59:03 2019 -0500

    Add ast.mli file

commit 72699b2195ca00f26e011fc3c9edc2c7d8341681
Merge: 14dbb41 1573290
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 12:59:26 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit ea8ca3eaccb36264852184a0c41b1df46a80b4c2
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 13:21:56 2019 -0500

    added operation with precendence to parser

commit da4f6479e425a1f423c92e51f986f1eaf7f1d1ec
Merge: ea8ca3e 72699b2
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 13:22:02 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit d87d25fed346976b892ee1445daad643604b8a55
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 13:23:43 2019 -0500
```

```
    removed comment

commit 96a4e989279c03c15b010fdeed38c4d441d522c2
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 13:25:47 2019 -0500

    removed dollar

commit 4d02cb925f70e80f6815be88343e8758f9588bb5
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 8 13:48:01 2019 -0500

    Fix tuple constructor

commit bf16a6bae840b9e1b67554199866ba13f4f0168a
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 13:53:04 2019 -0500

    Remove ast.ml

commit 54b0027330ecdbec12c8ff468a73da70a8a6da64
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 14:03:19 2019 -0500

    Add negate operator

commit 556b919560747e28bc6064fe62475bea879760dc
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 14:03:56 2019 -0500

    Remove extra Neq

commit 0c0dc472d617812f3085cfb999c818cb0c189edd
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 14:12:29 2019 -0500

    Added literals to ast.mli

commit 7d36deaa740c73620fba1a84016d17d41ad5fbb6
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 14:15:12 2019 -0500

    added of... to ast.mli

commit 4a439b45fc9861ad5af7d9f3c78dce99a3e2f86f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 8 14:20:40 2019 -0500

    Break up a line into two

commit 9dc2b9026d14a99150de6f4fd657fae44e953acb
```

```
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 14:23:59 2019 -0500

    changed types to ocaml types

commit bdcdd2e4e4badbfb65204b40027de6d22f875e36
Merge: 9dc2b90 4a439b4
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 14:25:02 2019 -0500

    added ocmal types to ast

commit f680a9c21efec413ad3dedd67d5250e54e5783b9
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 14:26:13 2019 -0500

    removed TLit

commit 9d522b72273b249daba56034f552b8f9f6bde9cd
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 14:31:47 2019 -0500

    Add entrypoint

commit 6c739e7900c47977a4431cb7857f74460e4b474d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 14:50:12 2019 -0500

    Add type annotations

commit 60586509d7f998c57f2bce09a67d2f3c9745ee6b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 14:52:19 2019 -0500

    Add not operator to mli

commit 82c557a115b4c7045fb84d748b9607815b594f02
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 14:53:06 2019 -0500

    Add missing comparison op

commit 507bfd88bfdc1b1b3109abaeb5b0eb1d4cbadb76
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 14:54:53 2019 -0500

    Added operations to parser

commit 08ade9dd59ccd408263682544e78b99151923100
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 8 15:03:06 2019 -0500
```

```
    Added the rest of the operators

commit 17f817a1ae01f09d6aa071750318172915a3d20f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 8 15:08:06 2019 -0500

    Fully qualify module

commit c3a6b11d30afb952a880dbc883dc5d8237dacab2
Merge: 17f817a 08ade9d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 8 15:08:09 2019 -0500

    Merge branch 'master' of https://github.com/al3623/rippl

commit a45cc64c659582ebf3f2419101262ab062b9d5ee
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 8 15:49:19 2019 -0500

    Add qualifier for imported module in parser

commit b398635dff5d7deeecb81be7dc59c62be842d8c7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 8 15:56:17 2019 -0500

    Fix assignment in ast and parser

commit 589e9af89e4419bbed61f2a7cc1f745a05b546c1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 13:45:38 2019 -0400

    Add diff to lexing tests

commit 10f4e3f92978e5ad450cd3e00ffe8c4885e82a7e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 14:01:38 2019 -0400

    Change clean target for lex tests

commit fbdc07e7399cfadc212ae1a179e3b60a41e7c11d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 20:44:18 2019 -0400

    Add literals to parser

commit c08017af87c811a777e5fd9b793c5aa4c73f278d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 21:44:28 2019 -0400

    Add list primitives
```

```
commit ec3587b270abb71185d4699416817ce3326d975e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 22:05:31 2019 -0400

    Make strings a list of chars in parser

commit 9b3a005683b3bfa2a06ba0750de1e531d97733fd
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 22:24:41 2019 -0400

    Add parser testing scaffold

commit 577288b0ccf2bfcc0e44c9cd027ddedaaf44109d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 22:36:06 2019 -0400

    Fix identifiers and lexer clean target

commit 0b3682dd947646b1e157bac19e6e6da26d13b082
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 22:39:15 2019 -0400

    Rename lexdriver to driver

commit 1142c52fcdc92fed200342158bdf6d3abd71452a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 22:45:20 2019 -0400

    Add IDENT to parser

commit e30c659510f5483849b0569466212aa48b2652d5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 13 22:47:42 2019 -0400

    Add first few parsing tests

commit b7ad555f065087c5d572100f079e25e882855db8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 17 17:58:04 2019 -0400

    Add preliminary list comp parsing

commit 720ce5780a6c4c9c9ddd5afbe2ad22d05d63938c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 17 18:14:17 2019 -0400

    Add clause and listcomp types to AST

commit 1c11cdbaec54890b9ca04b735cae3c550cb06770
Author: Amanda Liu <al3623@columbia.edu>
```

```
Date:   Sun Mar 17 18:28:21 2019 -0400

    Add constructors for other lists

commit 863172bf9b6eaf8f64c715c7d0a9324186fe713b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 24 22:31:42 2019 -0400

    Add compiling list comprehensions to parser and
    remove reduce conflicts

commit 780f61179e9785446bca930fa1cb48964c117079
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 24 22:39:15 2019 -0400

    Add ITE parsing without conflicts

commit 2d772a3ad873c80083901879b5df43b0ed6bc809
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 24 22:40:45 2019 -0400

    Add let bindings parsing without conflicts

commit 1be6085b4779e4a8ec90cc2a19d8d65de1682fe4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 24 22:42:06 2019 -0400

    Add parsing lambdas without conflicts

commit 15d891dd436f1e6014ba471c0e8846918fd21b02
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Mar 24 23:11:58 2019 -0400

    Add application operator to parser

commit 85ae14b6dd2a25fabf871cd0c78955d8dbe0980a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 24 23:42:17 2019 -0400

    Add application operator to lexer

commit 5d98c4a0eb141031a90f3ea770a468578f908630
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 24 23:42:47 2019 -0400

    Add application operator to lexer epilougue for testing

commit c2ce644552b8e23a162ac113f7364965da146b25
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 24 23:52:28 2019 -0400
```

```
    Fix hanging comment

commit bcde01d377fabe129aaabc75279e559aabc15a46
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 25 00:28:54 2019 -0400

    Increase precedence of application

commit b4732d223e5fbc2fd67fddee27c4e813c4a24886
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Mar 25 01:09:49 2019 -0400

    Add parenthesized expressions to parser

commit 17fc86128f10a0c62db6355fef6e0ee172332d7d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Mar 25 16:57:03 2019 -0400

    Add parsing for entrypoints as vdefs and type annotations

commit 688c843fe177d14e2c05587faa4ba959fab3e711
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Mon Mar 25 18:13:29 2019 -0400

    Update TODO

commit 8311643221004c50fed2c233629480934a7089b2
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Mar 25 18:44:52 2019 -0400

    Fix parser typos, add Neg to ast, wrote first parsing tester
    (most list functionality not implemented)

commit 015081c069ebc48e1687a3abba238f2560437818
Merge: 8311643 688c843
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Mar 25 18:47:29 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit c51f3b620e029c660a4812b64f01aee7a658b2fd
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Mar 26 01:29:59 2019 -0400

    change parse test printout format, fix parser typo,
    change parse test, add parse-test script

commit 93bc737598d984dfac2ca25e2082f69df47d88b8
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Mar 26 19:05:43 2019 -0400
```

```
    Update README

commit 203fe893a827f624617d2583d71dd6c0930758b0
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Mar 26 19:07:18 2019 -0400

    Update README with note

commit d85d5532c7bad9f52f834966ba891acf5b3a36a9
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Mar 26 20:21:01 2019 -0400

    Update README with parsing test

commit 68b84b8d893ea1f9f310199fb685566679e70c57
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Mar 26 20:25:38 2019 -0400

    Update README again

commit 60af3f82bca13020f63c0f67a739729377c735a7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 26 20:29:13 2019 -0400

    Change parens precedence

commit f62d084df4af50439bea01f6cf4e9d9bfe5c1c1b
Merge: 60af3f8 68b84b8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 26 20:29:25 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 1c88368a52c0ad9c5174362c1c0e9022a62423ba
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Mar 26 20:49:32 2019 -0400

    Add questions to README

commit 124bf628faa0eb88927904753f2be3e917046f95
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Mar 26 21:08:00 2019 -0400

    Update README with local function lifting

commit 0fcebc46ca5f18e45330ce8be84f248d3a54183d
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Mar 26 21:09:09 2019 -0400

    Add add parsing tests to TODO
```

```
commit 2d17aa6be8c3de3b91b07339f83a53700e1e8a44
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 26 21:42:56 2019 -0400

    Add check_main pass

commit 49b3b6237e07c76dd7aab2d90522caf27950441f
Merge: 2d17aa6 0fcebc4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 26 21:43:21 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 8041bc8300a6cb779f64078b86bb92d672bb8e08
Author: hlehv <hlehv@aol.com>
Date:   Tue Mar 26 21:50:28 2019 -0400

    Add check_main to epilogue

commit 63dedb61685b696547cde4271d6fc7611bf2376c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 26 21:56:26 2019 -0400

    Lift check_main

commit 7117a0eac849c8437b5e0e0d4f0a524e7e79f842
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 26 22:01:35 2019 -0400

    Add check_main symlink

commit d2153370e544932eaae4b5f9348671f722dcad58
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Mar 26 22:03:22 2019 -0400

    Add parentheses for epilogue

commit 778f0d98f8a59898bee2044fffa78d499bf32332
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue Mar 26 22:10:56 2019 -0400

    Remove main token from lexer, parser, and lexer epilogue

commit 21ec16a5260dd75a1feadedd15e531f953d3fafd
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Mar 26 22:27:17 2019 -0400

    Modify parsing tester (epilogue) to take more than one Vdef

commit 1ab25571e1314cf6adda06b8ca86a86e5f899f81
Author: Hans Montero <hjm2133@columbia.edu>
```

```
Date:   Tue Mar 26 22:47:37 2019 -0400

    Add wildcard _ token

commit 3290b87427a8cb53fe581f5978f67d963ce708ff
Merge: 1ab2557 21ec16a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue Mar 26 22:47:44 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 0cbe65aa0dc0893d0f25c19b0b61ebcfe9d7b762
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 27 12:59:28 2019 -0400

    Make it so simple assignment is not an expression

commit 1988b6c6d3823f54ccc49f2a95489e0601f41e30
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 27 13:19:49 2019 -0400

    Make src able to build with compiler passes with Makefile

commit 1d0ad9958a37e7596c9c204502bb600d84532323
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Mar 27 13:24:17 2019 -0400

    Remove excessive pass from compiler.ml

commit 91d59408d9fba359f6f141f172d5e0478dc95c82
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Mar 27 13:25:30 2019 -0400

    Update README with assignments and finished tasks

commit 702f4ed17f3d61fd36cab2350b7dbbb0380bda66
Author: hlehv <hlehv@aol.com>
Date:   Wed Mar 27 22:33:18 2019 -0400

    Change parse-test to accept command line argument

commit d14190c466a89773e56add4624101cfe07be1049
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu Mar 28 22:43:56 2019 -0400

    Finish parser epilogue

commit a5813b7b2dd28154f137805291c00ff471bd097b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Mar 28 23:18:52 2019 -0400
```

```
    Start type checker

commit 0cc8dcf237237061677fcd92210536b15b3df68c
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 29 02:45:21 2019 -0400

    Added initial version of check_lists

commit 7ae8a47db62af7b30437d9a398ddff1d1473852b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 29 11:06:41 2019 -0400

    Add complete typechecking for everything except tforalls

commit 3eaae37e71f254d7727426aad793674611278657
Merge: 7ae8a47 0cc8dcf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 29 11:06:59 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 9be98364f2e32618f25cb45725f94a0534c72ab3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Mar 29 11:13:26 2019 -0400

    Make typechecking temporarily compile with partial

commit 7a3e341c588705c5b055aaca6f40d956a4e5d82b
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 29 12:41:36 2019 -0400

    Added list comprehension clause check to epilogue

commit 4e8631c1eb0a53887f18eb18ed6060f01d3708f2
Merge: 7a3e341 9be9836
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 29 12:41:42 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 053009cdbbfc2a6e0b2d232ef4596fecf1856717
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 29 13:14:59 2019 -0400

    Added check_lists test to passes

commit 9f64775d39bc10ca5b130df92f960c188da99423
Author: hlehv <hlehv@aol.com>
Date:   Fri Mar 29 13:47:37 2019 -0400

    Added check_scope function to check if variable has been defined
```

```
commit 94684b295efb6eb19aa8e3d452af8a13c68b0c87
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 29 14:08:29 2019 -0400

    Add annot vdef pairing semantic check and make utils
    directory with pretty printing for types

commit 761e64fd2145fc9b34ba8af6a4d44ad167fdb7ca
Merge: 94684b2 9f64775
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Mar 29 14:10:08 2019 -0400

    Resolve conflicts between list comp and annot semantic checks

commit e42c03bd57d452edc61cb9a594d490f7a1ce8e89
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Mar 30 20:33:56 2019 -0400

    Add typed ast to ast.mli

commit c111f8c634ef54c71ed90565b7864e55993862dd
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Mar 30 20:34:27 2019 -0400

    Move typed ast to bottom

commit 75fd14e4793a63e9feb15d35b780cd3c686dfb86
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Mar 30 21:34:02 2019 -0400

    Add typed type system for hello world

commit 948aba842d2e07a54e9f02ef6f1a1e0a2c028028
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Mar 30 22:36:05 2019 -0400

    Move assign out of expr part of AST

commit 477e628e1b394e2fcdf789d3b0b1b3af24e47b9b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Mar 30 23:26:25 2019 -0400

    Add ty to tuple in typed ast

commit a6d74bf57d172aa849d2a3b3646032944802bd7f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 31 11:20:14 2019 -0400

    Add compiling codegen.ml for hello world
```

```
commit 940883f8f68d329a6db032cac6d8e28f47b20cb3
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 31 11:20:44 2019 -0400

    Use boolean equal instead of string equal in semantic check

commit dd0f890139856eb41796d84e1b82d5d939f7f906
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 31 14:07:23 2019 -0400

    Run LLVM using LLVM interpreter instead of compiling to x86

commit b8ff88558e784801e6a3a2c6c8247fa7009c709c
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 31 14:07:45 2019 -0400

    Slightly enhance hello.rpl

commit 9a998b8fc0d6330d81c94be4f05d08882a77cde8
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 31 14:32:00 2019 -0400

    Update README

commit 0d252058ac800dc5a5fa4b6e93d33ab8c3962379
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Mar 31 14:37:38 2019 -0400

    Clarify llvm requirement

commit a68ec3b639d3265a7010181c357b57b3abf6ba72
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 1 16:13:05 2019 -0400

    Change list based parsing rules to support empty list

commit d024d6c9a039b87be0558dab73c5918faee7f1eb
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 2 18:14:25 2019 -0400

    Add Tast file and make hello world work

commit a389e0f302f4d1a775e4b7893b16ee7e20bf4f8b
Merge: a68ec3b d024d6c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 2 18:14:44 2019 -0400

    Merge branch 'master' into rippl-dev

commit 88d414787d56d482b3654b169d23c4a15327edcd
Author: Amanda Liu <al3623@columbia.edu>
```

```
Date:   Tue Apr 2 18:17:09 2019 -0400

    Fix assignment parsing to string not expr

commit 205963723f36850634c92fc99f3f4ea9e1df7b81
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 2 18:19:14 2019 -0400

    Match assignment change in parser epilogue

commit 890c76ff6c56292f183dde6f53cb4870be4b7ff0
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 3 13:13:34 2019 -0400

    Add TODO file

commit 8b9ccc03fed5f7bfa7d61b884854adfe138f264c
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 5 16:00:47 2019 -0400

    Add get_fresh_var to utils

commit 971e1c893ec0c87e8defa7124424580bb7e11a98
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 5 16:15:40 2019 -0400

    Actually rack get_fresh_var

commit abdaaf11085110d015bcf90055343959 78a4f611
Author: Hollis Lehv <hml2138@columbia.edu>
Date:   Fri Apr 5 16:20:29 2019 -0400

    added parameter to get_fresh_var

commit 44745d348b8a95ab63779f80ba0476fb5dc30076
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 5 16:25:39 2019 -0400

    Use get_fresh properly

commit cc260a72ec34ea0bff043288a75361814dbedbe6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 5 17:49:53 2019 -0400

    Create c library file to be called from codegen

commit 00412539ac03ad4006dc314f8f3dd2b9c20e37d7
Merge: cc260a7 44745d3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 5 17:50:09 2019 -0400
```

```
    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 6e3e8298f2cb7303b9571880147fb707b9c78c55
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 5 18:31:49 2019 -0400

    Let compiler native take file name and produce executable
    and link against lib.o

commit bf80b1a8b7ef717f6670c10d90dfa1e3961c43a4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 5 23:24:07 2019 -0400

    Add void * equivalent and struct generating comments

commit 833f3ae3269d704945e5c5398ac558fed6c2b7cf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 5 23:24:40 2019 -0400

    Add lazily eval functions for list ranges and infinite lists

commit 01fa0c7295824d816da2980b75929c2cf311174b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 5 23:52:56 2019 -0400

    Add empty list constructor in lib.c

commit 346e334de89421459680c4370f1e7d08ac2a92ed
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 6 10:24:49 2019 -0400

    Change compiler to report non-zero exit codes

commit 49e3b41e4a74ef3a70e3b4e57e53b2fa752be79e
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat Apr 6 16:44:17 2019 -0400

    Identifty lambdas and variables inherited from closure

commit 911734ca638e4fa687f2c4c4c8ec5d3666965eef
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat Apr 6 18:11:00 2019 -0400

    Identify lambdas to be lifted and variables required for nested closure

commit 2a4ffffe9d90617c49503a0e453bddad5f1f1052
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat Apr 6 18:21:17 2019 -0400

    Ignore top level lambdas
```

```
commit fa6d8ff8070f562b6392127b694e0652fb0fe766
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 6 18:40:43 2019 -0400

    Add more lib functions and separate function declarations into lib.ml

commit 2d967f0f8a9a787e6ee9041c361bf9b511a0ccb5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 6 19:10:40 2019 -0400

    Make struct ListComp a non-opaque type

commit bcf632de13b6fcdf5c28607190628e5033cf76b6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 6 19:41:00 2019 -0400

    Lazily print and evaluate list ranges in main

commit 21180c827f6078333fd7d5941284d7c0e6fcaee3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 6 20:05:08 2019 -0400

    Add support for empty lists and fix temp inferred type of lists

commit f0b2e904014afabff84c5df6f5a52788cdd56086
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 6 20:48:01 2019 -0400

    Make hello world work with loading list literals

commit 6a59cf49f1e77a312b52bfb4e4b5d2935fdeb928
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Apr 7 00:11:53 2019 -0400

    Give anon lambdas names, mangle all lambda names, create closure map

commit 18a8fa62302b9986a70abbbcc039a7e5b63b5e7a
Author: hlehv <hlehv@aol.com>
Date:   Thu Apr 11 23:03:36 2019 -0400

    Modified compiler for type inference printing

commit 778cb582f0502a6f002e7503be7a66ea4bce0317
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 12 14:58:49 2019 -0400

    Give application more precedence than rarrow

commit dd364e41ff60f5882eb820c7cd9373d7cc176e0e
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 12 15:04:21 2019 -0400
```

```
    Account for scoped variables in closure

commit 8c4fb9b88d3df2ebf1da2638fd829c30672d8eb2
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 12 15:39:43 2019 -0400

    Add testing suite for closure

commit a342ba73e13caeb82b40c3e4c561cdb9099730c8
Author: hlehv <hlehv@aol.com>
Date:   Sat Apr 13 17:20:15 2019 -0400

    ftv and apply added to type-inference

commit 2a979840411125945a2de208ab3275150b93a624
Author: hlehv <hlehv@aol.com>
Date:   Sat Apr 13 18:31:55 2019 -0400

    generalize, and helper functions for env's work

commit a34a2ee0ec1a3cfa81f95e78afb2851db0850eed
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 13 22:28:42 2019 -0400

    Change parser epilogue to print var for application

commit 185369175b452f56bd32eb34f5c174f108407d23
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat Apr 13 23:01:47 2019 -0400

    Set up code to mangle and replace, need to commit to test on CLAC lol

commit 6b0b55af3cd8c10dedf3cc75d922c8be5d0c07fd
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 13 23:16:17 2019 -0400

    Rename list struct

commit edaab8714d8e3d7f523678181dcbb7e13078708f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 13 23:34:10 2019 -0400

    Complete list rename and include comprehension fields

commit 6556b49fb3e4559cc0ba336bad68650e7e850a68
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 13 23:51:40 2019 -0400

    Add new comprehension fields to codegen
```

```
commit 1cbeb4cd993cc484857dae1c0b0eb0a0ac525708
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 14 00:00:37 2019 -0400

    Add filter pointer to array of function pointers to comprehension struct

commit 435dfd877416c58b3f84e95326277882d34bc36c
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Apr 14 00:35:51 2019 -0400

    Fix minor bugs in mangle-replace algo; also no need for CLAC

commit 66e308fbf3e9a5b50d56760a16b86fc24248ca18
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Apr 14 01:04:46 2019 -0400

    Set up scaffolding for actual lambda closure

commit ee94f0b8fcad56c48b51b8cc8c48366e66e808cd
Merge: 0d25205 1cbeb4c
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Sun Apr 14 13:03:40 2019 -0400

    Merge pull request #1 from al3623/rippl-dev

    Add more functionality for list comprehensions and "Hello world"

commit e9a746e183beb73ccfcdc3374be2dde83115a9aa
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Sun Apr 14 14:37:13 2019 -0400

    Change make<Type> functions to return i32_t, add i1_t type,
    add ITE case to type inference

commit 0cb4acf491ab2e1e7f05afd29d01e2563177eb72
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Sun Apr 14 14:39:53 2019 -0400

    Add codegen for basic if-then-else expressions, add simple ITE test program

commit b4bd4847a142a88647804bf6a12f7dbb29b45386
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Apr 14 15:42:19 2019 -0400

    Close lambdas

commit 9d160d48a8d60e7948af9d74370cafc3f61a6cc8
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Apr 14 17:13:57 2019 -0400

    Lift lambdas... finally
```

```
commit 56dac29c3aced81d2581d1cb759b09c9225b25f1
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Sun Apr 14 18:02:11 2019 -0400

    Revert lib pointer types, add i32_t/float_t to i8_t
    conversion functions to C lib

commit 8fda5824d74908789b11a84e8f8f5083a1760a32
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 14 18:30:30 2019 -0400

    Update struct comments in lib.ml

commit 08ecc113afbb655d8af134b7b904cded44c4906f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 14 20:53:59 2019 -0400

    Add more fields to list comprehension

commit 5a4662f3621a788f134f2823233bc8e7708b0352
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Apr 15 15:36:55 2019 -0400

    Support list traversal for free var identification

commit f807d3fcc8c0dd3e6ef23dd96e99026ea3638d5d
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Apr 15 16:30:04 2019 -0400

    Add another test

commit cdda7909c00d4b058cd5c63913663da2ddf71bbf
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Apr 15 16:46:36 2019 -0400

    Support lists in free variable detection

commit 1cd81e93d28be218ff25fa475ca837604f5e4bea
Merge: cdda790 f807d3f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Apr 15 16:47:00 2019 -0400

    Merge branch 'lambda-lift' of https://github.com/al3623/rippl into lambda-lift

commit cb926f4ecb8ed78883d1d460ebf8cfe372c0b42c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 15 17:34:37 2019 -0400

    Add thunk representation idea
```

```
commit b59a3374aed924a0a870ab6cad442b6c3fa2543b
Author: hlehv <hlehv@aol.com>
Date:   Mon Apr 15 19:02:14 2019 -0400

    Modified tast to take subst, mgu and ti

commit ac6a91d5aeafba98b9536911fbea1202e96565d6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 15 20:11:56 2019 -0400

    Add working thunk demo

commit 357339af8910047608a805cea7e26a915822686d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Apr 15 20:26:12 2019 -0400

    Mangle let bindings in lists

commit 050c77e33c273204ec1d9921ba52f5860e395ba2
Author: hlehv <hlehv@aol.com>
Date:   Mon Apr 15 21:41:16 2019 -0400

    Added tast pretty print

commit 5ad8bc349c614611f95e138de085d9e5e011e11d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 15 22:04:34 2019 -0400

    Start remove subst pass and add iast representation (inferred ast)

commit c1f2ed18063e415fc66a2f4cc5fa4db0a810a818
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 15 22:09:12 2019 -0400

    Add remove subst call to compiler

commit 886e64b365ca74500c6cd5540867a43d7c25f991
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 15 22:17:52 2019 -0400

    Fix small typos in remove_substs

commit 5a207c2618447fe1030e89b2d4f989155d54c645
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 15 22:45:29 2019 -0400

    Add thunk llvm library

commit 6c7d8a6cab248965b84760d895ae53101feb23ae
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 16 12:51:00 2019 -0400
```

```
    Add module to be included for miast map

commit ad0175235d6413f36ee3eb2e2bfb6dd07c343ffe
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 16 15:52:08 2019 -0400

    Add inference support for list literals

commit d5ab8ead35d0933f50de0f7bf241c4ff453f734b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 16 15:53:57 2019 -0400

    Reformat inference code for style

commit 5139473b48cd3cff726a34f06dbc972b0cd6e854
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 16 16:42:29 2019 -0400

    Temporarily pluck main lambda to test inference

commit 749ead709747ef75990a957abf33984bb63c01d1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 16 17:04:03 2019 -0400

    Add scaffolding to print types and test inference

commit 1cb2e64410c242b4cd7460a2699135254691e393
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Apr 16 21:01:04 2019 -0400

    Refactor part of codegen, change makeBool to take 1-bit param

commit d4e138bb0bd619f6b0ab4f71c6a0845e08003e83
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Apr 16 21:43:19 2019 -0400

    Change printing method for int/float/char,
    remove unneeded functions from lib

commit 0f4edf6c35baff4fcfba49a5250264c5ecd1b786
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Apr 16 22:36:58 2019 -0400

    Add codegen for some lists

commit 57c251d0f68a82c9db3de51b5bb957a14d4c9662
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Apr 16 23:31:18 2019 -0400

    Move list printing to print_texpr fn,
```

```
    add printBool fn to lib, only print main

commit 6b7f78068b7371f9855125a9a3ff657f442d4444
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 17 13:27:09 2019 -0400

    Update TODO with anticipated conflicts for meeting

commit 07031f5b568566c71b4427798792e71bfbb724ca
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 17 15:10:54 2019 -0400

    Lift lambdas in list expressions

commit 8508829a93dfc5becdd3a07921b0f59de563f97e
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed Apr 17 15:36:08 2019 -0400

    Add skeleton for codegen for lambdas

commit c8f27adf6b437ee10abbf676901e86e45e18a3b3
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 17 15:49:24 2019 -0400

    Comment up lambda lifting code

commit e829693a74280f8109daa4597d6ce6c10a88ec98
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 17 19:15:48 2019 -0400

    Remove debugging statements

commit 8645f9391622cf5813c306b3d17d6ce9293814c9
Merge: e829693 6b7f780
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 17 19:32:54 2019 -0400

    Merge branch 'master' into lambda-lift

commit 18734f9b2038d342fb57711a441b892e9cba0a20
Merge: 6b7f780 8645f93
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 17 19:34:38 2019 -0400

    Merge pull request #2 from al3623/lambda-lift

    Lambda lift

commit d2e7025a82709e94dce23ebf3f22184d5b75882c
Merge: 8508829 18734f9
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
```

```
Date:   Wed Apr 17 19:44:42 2019 -0400

    Merge branch 'master' into rippl-dev

commit 525b0f598a65ff568f85fc989003337ccb2f7812
Merge: 18734f9 d2e7025
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 17 19:45:43 2019 -0400

    Merge pull request #3 from al3623/rippl-dev

commit 11b793a9f70296e0d2792c50ad63d50eead608c5
Merge: 5a207c2 525b0f5
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 17 20:19:51 2019 -0400

    Merge branch 'master' into list-comp

commit 2523f57ecede630758c7e13197ec116793244345
Merge: 525b0f5 11b793a
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 17 20:20:51 2019 -0400

    Merge pull request #4 from al3623/list-comp

    List comp

commit 03a088cf38e44d042e1e0a4fc28faa33f88be452
Author: hlehv <hlehv@aol.com>
Date:   Wed Apr 17 20:23:33 2019 -0400

    moved 0 over

commit ba368cd8ed7d493b38f693aead084b0f7b1666ce
Merge: 03a088c 11b793a
Author: hlehv <hlehv@aol.com>
Date:   Wed Apr 17 20:23:49 2019 -0400

    Merge branch 'list-comp' of github.com:al3623/rippl into list-comp

commit ffe4ef94144aec8d1db775e60e2a9158178d8bd2
Merge: 749ead7 2523f57
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 17 20:43:47 2019 -0400

    Merge branch 'master' into type-infer

commit 69e1d1bc9649706d3fed0c305c9157e122a40c99
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Apr 17 20:47:28 2019 -0400
```

```
    Add lib to Makefile make

commit d2b2e9cf59463623b80dcd2bf4f819bdb5d098f0
Merge: 69e1d1b ffe4ef9
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 17 20:48:50 2019 -0400

    Merge pull request #5 from al3623/type-infer

    Type infer

commit a0b92adaaa21d1174fd381a881ea6d0314721897
Merge: ba368cd d2b2e9c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Apr 17 20:50:14 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into list-comp

commit b09b00f46b9aa7ee21b6e3d46829465ce20842ac
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 19 13:46:30 2019 -0400

    Fix typo in compiler

commit e4731be830b78591d9c0d04d4710e081016b8605
Merge: a0b92ad b09b00f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 19 13:48:07 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into list-comp

commit 6c78a29fbfd8c399b68db8b992318b3a405f3dbc
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 19 14:39:40 2019 -0400

    Get rid of compiler warning

commit 967d5163e5196be90c2a353b49595ede9d9765e0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 19 15:12:55 2019 -0400

    Refactor C library and add map

commit a233746aba15f490149973165ea27ef2f6e27e58
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 19 16:42:23 2019 -0400

    Add map and filter implementation and rename thunk invoke to apply

commit f417f7640011b4f1065e5198c10592d25debceb1
Author: Hans Montero <hjm2133@columbia.edu>
```

```
Date:   Sat Apr 20 00:27:38 2019 -0400

    Change list comp to take a string; wrap comp constructor in lambda

commit 1cb111508ffa4f9c6adfacdd721837e3d4a7e2cc
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat Apr 20 16:03:19 2019 -0400

    Wrap list comp constr in lambdas, ensure closure, and lift

commit 000ab2c9b92ab9be0d9218634c8a6a6cec293f2f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 21 16:16:11 2019 -0400

    Create filter proof of concept and rewrite list
    representation to wrap thunks

commit 4bd08445314406f6aea569049bbcf9c19391a8ac
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 21 16:41:07 2019 -0400

    Add working map proof of concept

commit 686a14653fde3fa10a3a4b5c87e80356693ca9d2
Author: hlehv <hlehv@aol.com>
Date:   Sun Apr 21 16:44:12 2019 -0400

    Added function to type-inference

commit bb5c1e703a7d2067b21ad0e545773f148d84c5e9
Author: hlehv <hlehv@aol.com>
Date:   Sun Apr 21 16:45:23 2019 -0400

    Commented out beginning of let

commit 331e64d9633e7a75c5e9a5be6f160a951c552271
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 21 18:28:46 2019 -0400

    Add inference/checking for list ranges and infinite lists

commit 7ed794934065b36c2754bf1e6213ac64f26f8410
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 21 19:12:22 2019 -0400

    Make lambdas and listvbinds take strings not exprs

commit 847ca367f6dbd3d3bd9f17dbfcccd0d83d71544b
Merge: b09b00f 7ed7949
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Sun Apr 21 19:14:48 2019 -0400
```

```
    Merge pull request #7 from al3623/type-infer

    Type infer

commit d81432d0cf02c4872f7d320e463e72bf9d5a656c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 21 19:16:48 2019 -0400

    Start inference on comprehensions

commit a802edd6229e9c78f485b03dc51d2d53fda94e15
Merge: 4bd0844 847ca36
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun Apr 21 19:17:07 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into list-comp

commit 5215e7a34efe66a88fb17a60a5f77870dc430135
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun Apr 21 21:03:05 2019 -0400

    wrap filters in lambdas and lift them

commit c59b48e1fdb773523c316135b962a3d3f31e85ee
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Apr 22 00:34:29 2019 -0400

    Implement textual replacement in list expressions

commit 2d9af814c8a7c8ccd6af06ef6237ea846f5b99a8
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Apr 22 01:21:50 2019 -0400

    Add pre-mangled lambdas to map to ensure uniformity

commit e7fa2beb97d66d52201ab44dacde14027484dfc5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 22 16:16:10 2019 -0400

    Add codegen declarations for map and filter

commit 51f8a8a1d31453351564a12440ef824f6b00c045
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 22 16:41:16 2019 -0400

    Add inference rule for lambdas; to be tested

commit 939659deefaa4906b19114dc1c82fa75df93234f
Merge: 847ca36 e7fa2be
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
```

```
Date:   Mon Apr 22 17:11:33 2019 -0400

    Merge pull request #6 from al3623/list-comp

    List comp

commit 29146ac36b2108298bbed54e445764dee6ed05d8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 22 17:14:16 2019 -0400

    Add inference for application

commit e5bb5f048a61c973093c7c482c2cf68f8c713427
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Apr 22 17:25:48 2019 -0400

    Change lambda to take a string instead of Var(string)

commit 3da12d14aa4901b87e00fc2438a22ef61ea2ca66
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon Apr 22 17:27:21 2019 -0400

    Parse lambdas differently and change precedence of boolean ops

commit 8d3e3ac08239bc3b59c2a1b0f104b5fe657cd369
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Apr 22 17:41:10 2019 -0400

    Adjust parser test for ast changes

commit a92d623dc166e2a24094f3937b4b183cbb47cb8c
Merge: 8d3e3ac 939659d
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Apr 22 17:41:22 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into rippl-dev

commit 6b8f94662c2c0626493c77c3c5e80ad066a8de68
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 22 17:42:40 2019 -0400

    Add type merge error

commit 51634e3fd62ba5cc8c03bc3e59943bd32e3c95c2
Author: hlehv <hlehv@aol.com>
Date:   Mon Apr 22 17:46:28 2019 -0400

    Added operators to inference

commit 60410d493d5d4759720e5e385dd02f0a71921cb0
Merge: 51634e3 6b8f946
```

```
Author: hlehv <hlehv@aol.com>
Date:   Mon Apr 22 17:46:41 2019 -0400

    Merge branch 'type-infer' of github.com:al3623/rippl into type-infer

commit 5b2a5c66f91dd94c611327823270dc4b3878af42
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 22 18:20:48 2019 -0400

    Fix epilogue pattern matching with strings instead of exprs

commit 07bc23af5a2b0eea37c2ce118bb210d4c34637b0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon Apr 22 19:29:25 2019 -0400

    Fix apply for inference

commit 92a6038fd70bbfec2c4173c759b5fe0c2c22cbc3
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Apr 23 13:09:36 2019 -0400

    Change app operator to ~, change epilogue of parse test
    to print Application correctly

commit 7b37f2c6224a8554a1c49c9a0a6fe6ea2780cb3f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 23 18:22:57 2019 -0400

    Change application operator to tilde

commit 05476830661998ab5d00de06b1c810a85cd2f574
Merge: 939659d 7b37f2c
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Apr 23 18:30:04 2019 -0400

    Merge pull request #8 from al3623/list-comp

    Change application operator to tilde

commit 2b139c197cd92b30fc75c3b8ba53453df1e36d18
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 24 01:56:29 2019 -0400

    Rework wrapping stage, consider top level scope,
    move lift code out of compiler

commit 365820f66d252c65a3278b264c2adb4bd3808615
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 24 02:08:40 2019 -0400

    Group together consecutive lambdas
```

```
commit ccf98e54926bd08914ef304284397d31c4384b8f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 24 03:21:55 2019 -0400

    Remove legacy wrap code, support lists again

commit cdb2b34084d4b588ffbf0b40edb27dcfef90c608
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Apr 24 18:38:05 2019 -0400

    Redo pretty printing

commit 862032d0a80625d56be03698cc0fba8a865fb4c9
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 24 19:15:33 2019 -0400

    Still working on replacement

commit 3764e4a2b73c157e4624b8124123aa6d93cb725a
Merge: 862032d cdb2b34
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 24 19:15:50 2019 -0400

    Merge branch 'lambda-lift' of https://github.com/al3623/rippl into lambda-lift

commit 93a0cb3a68be14abbfd0e3aa8325f1ec45070d7a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed Apr 24 20:22:54 2019 -0400

    Enforce consistent lcomp wrappings, finish up general closure

commit 425e5cdc14e539f4c52f1b5c475deb04b91ef2b1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Apr 24 20:24:23 2019 -0400

    Add possible indented pretty printer

commit 9a74b441de0aa092ae926495982373e08656dd11
Merge: 425e5cd 93a0cb3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Apr 24 20:24:39 2019 -0400

    Merge branch 'lambda-lift' of https://github.com/al3623/rippl into lambda-lift

commit 963a939ecfb6a636a2a744c08d5b56a00ca5a02b
Author: hlehv <hlehv@aol.com>
Date:   Wed Apr 24 21:14:23 2019 -0400

    Added let bindings to inference
```

```
commit a28ae01221d2ae549c0ac5db1848c18b7043f2be
Merge: 9a74b44 0547683
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed Apr 24 23:29:55 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into lambda-lift

commit 71006d08bd0b20f6966240d175eb72dda25ee01f
Merge: 0547683 a28ae01
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed Apr 24 23:48:57 2019 -0400

    Merge pull request #9 from al3623/lambda-lift

    Lambda lift

commit 472886fa40f6bfc544bfb60eb86a5a328004eab5
Merge: 963a939 71006d0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 11:33:39 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into type-infer

commit c516b42a20463c935f5ec41045c0071ad344d8c3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 12:58:32 2019 -0400

    Add list operation inference and ITE inference

commit aaf8ec241cc1ae8ac5cc3ff9b579110eafb2219f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu Apr 25 14:36:26 2019 -0400

    Change variable name in repl_body

commit 7175e94a1fd255c9a9aed347ced3356160b4620e
Merge: aaf8ec2 a28ae01
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu Apr 25 14:36:38 2019 -0400

    Merge branch 'lambda-lift' of https://github.com/al3623/rippl into lambda-lift

commit 555c7a2f61aad8be90b0f3e8928227d14ca50235
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 15:15:23 2019 -0400

    Fix Ite typing and extra space in pretty printer

commit 6e0da67bdbde55d250108360a7203c429eacffab
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 18:10:26 2019 -0400
```

```
    Add support for inference between vdefs and inference test directory

commit 485410aa06d7e7e7c9f92c5b1da133035bf58a22
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 18:49:21 2019 -0400

    Change lambda closures to support recursive lets

commit 249eaa7f065fb6fa70b6aa1f396885d6bf643500
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 19:36:49 2019 -0400

    Add modulus operator and inference tests

commit 4b2ef930932c54202c69ddca033103eb43ba305d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 19:59:51 2019 -0400

    Prepare to add tuple, maybe, and list operators

commit ae90304a2c3ac13d023c692c8009e08a8fa3f444
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Thu Apr 25 20:08:57 2019 -0400

    Update TODO.md

commit e013ddb65d54599e61dceb6a1e5070efbbafec29
Merge: 4b2ef93 ae90304
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 20:09:23 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into type-infer

commit 454e39b21b2ac873684635cd44c65bc490acc840
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu Apr 25 21:52:58 2019 -0400

    Add list operators to implement in lib files

commit ad16ab063a8d05695381b3f26c19b365c0b2c003
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 10:50:49 2019 -0400

    Fix compiler warnings in lib.c

commit f43405e8cc645f8a1ec48ee08a258dedaa38e8e1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 10:54:42 2019 -0400

    Add parsing for constructing tuples (no conflicts)
```

```
commit 8bf6338795b2b5aa7fad105162c8fbe5a5875119
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 10:56:31 2019 -0400

    Add first/sec operators to parser

commit 5d8ba4ac0daba75baee08b839555d1efb435912c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 11:09:18 2019 -0400

    Change nothing to none and add maybes to ASTs

commit 8e19ce9a188dc4777394a5408036828fd3554f16
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Fri Apr 26 11:12:09 2019 -0400

    Update TODO even more

commit 76b087a523cd0a49f32a9a5a37cae1dfbf6a25aa
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 11:19:52 2019 -0400

    Get rid of one level of indirection in length implementation

commit 403ba3621051c8cbbb52b875a3af95fb48cd0e45
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 26 11:48:16 2019 -0400

    Simplify compiler

commit fc9c722657387054cd816df913c4af291a9d9097
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 26 11:53:40 2019 -0400

    Comment out code from lambda lift

commit 961d1217a011c68447736296f989a3c82c3f89f2
Author: hlehv <hlehv@aol.com>
Date:   Fri Apr 26 12:39:20 2019 -0400

    minor changes

commit 293dc94e873ada44368807061852d4dd42a25379
Merge: fc9c722 e013ddb
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Fri Apr 26 12:40:34 2019 -0400

    Merge pull request #10 from al3623/type-infer

    Type infer
```

```
commit 85f00ae84d8bea14edbe6db40b9ae3e9c551b1fc
Merge: 961d121 e013ddb
Author: hlehv <hlehv@aol.com>
Date:   Fri Apr 26 12:40:39 2019 -0400

    Merging

commit 0953c7056fc71a1198b2bfebbe408053a7b39f33
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 26 12:40:51 2019 -0400

    Comment out debugs

commit 9947a75c288d3444d67893835111a413c4f31982
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 12:41:28 2019 -0400

    Add more implementation for lists

commit 8ea49db4ab9816ebbbbb7f80cbadb5a39d67360b
Merge: 293dc94 76b087a
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Fri Apr 26 12:52:47 2019 -0400

    Merge pull request #11 from al3623/list-comp

    More language native operators and parsing stuff

commit 21dd4dc6c6c5ab22ba61f8867333e58012a7b94a
Merge: 92a6038 293dc94
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Fri Apr 26 12:53:53 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into rippl-dev

commit 525c3ebc96aee3de969d3b6de28b101ce6daa4e5
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri Apr 26 13:15:41 2019 -0400

    Remove map and filter tokens and adjust precedenceof APP

commit a9adf3e26ba71542fe4f34028c13cdedf72fb90f
Merge: 525c3eb 85f00ae
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 16:07:51 2019 -0400

    Merge branch 'type-infer' of https://github.com/al3623/rippl into type-infer

commit cbf847173ced5205c191fdfffa23eccee47613da
Merge: 9947a75 525c3eb
```

```
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 16:08:35 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into list-comp

commit 8ea71fd9ce23e1ec6151df6cdedd6e0380d4a288
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 16:17:56 2019 -0400

    Add inference for tuples

commit cc40b1316241d1408c41f8d60aedae89f8c178ef
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 19:02:42 2019 -0400

    Add eval function to thunk

commit 48416a98971ea2a5e9fbaa660cc30eda787fa187
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 21:34:37 2019 -0400

    Make new thunk implementation changes to thunk.c and thunk.h

commit 6ab038330b0e1d9242f99351b8672666273916f4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 21:37:40 2019 -0400

    Change signatures in thunk.ml

commit 0faf162106ceb8fb673fd4db86cf83fd71ef9e70
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri Apr 26 22:34:27 2019 -0400

    Remove unecessary f function pointer form thunk and functions

commit 2e81979d6c9626f9bb6f688430d55210def83d2a
Merge: 21dd4dc 0972d80
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Fri Apr 26 22:47:48 2019 -0400

    Merge branch 'list-comp' of https://github.com/al3623/rippl into rippl-dev

commit 1280332897380e3504bc928d39383b144943ad59
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 27 14:36:06 2019 -0400

    Rewrite map and filter to support new thunk structure

commit b85a35afb711151658de2700342c2801d1289ec9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 27 15:04:10 2019 -0400
```

```
    Start list map and remove one level of deep copy in filter

commit f72745798ffdbfe6858d4ea838831d11a94b8d41
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 27 15:52:43 2019 -0400

    Implement cons, cat, head, tail, length in lib.c with tests for immutability

commit 2f3fa2f7cff9b518b744fb2ba61a666976338782
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 27 16:51:00 2019 -0400

    Implement map of list on list and example in mymap.c

commit a89a304c1eb654729b395c5ad5e73af50db2176d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 27 17:14:16 2019 -0400

    Add map_list declaration to mymap.ml

commit 9998e38d0f6ec78493a2c33ef16bf4248a5123b1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 27 18:21:27 2019 -0400

    Fix bug for applying/invoking thunks

commit 0972d80468fbf0f62e051b51cb1cea0f9b9acddb
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat Apr 27 18:22:27 2019 -0400

    Remove wildcard from main

commit 0c046e22c051eeb581ddd57c7f04091951ba55f9
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Apr 29 14:04:39 2019 -0400

    write some Codegen framework

commit 889940d034ae01d3ee85fc6f0f363b1434f620c4
Author: hlehv <hlehv@aol.com>
Date:   Mon Apr 29 18:55:06 2019 -0400

    Began work TI of clauses

commit c48ba7968b3c63a5f9c5e3e91abc1d5c50e1559d
Merge: 525c3eb 0972d80
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon Apr 29 20:53:05 2019 -0400

    Merge pull request #12 from al3623/list-comp
```

```
    List comp

commit e6aec8d85a173d66cd65e8d33a3e5a7057c6bb39
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue Apr 30 15:03:06 2019 -0400

    First lambda fold in codegen, fix whitespace in lib.ml

commit 659464d3558b6105efb5a443bfe1d77dfd6daed9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 30 21:58:45 2019 -0400

    Fix inter-vdecl inference so substs are consistent

commit 74b6b9610c5a0ead87a7ac8872d5a09244712373
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 30 22:05:59 2019 -0400

    Remove debug print statements

commit 66181b781a832c58899cf364a6bbb9f2ffdbf24d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 30 22:17:49 2019 -0400

    Add inference for tuples, maybes and tuple operators

commit ee14c3ce3783c31a5bcabebe5ee1e54d98e4efc6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 30 22:28:55 2019 -0400

    Add cases to remove subst in pass and fix typo in ast representations

commit 4ce8d1699b78a5455845650a0188c85f2f326cd2
Merge: ee14c3c 889940d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue Apr 30 22:43:22 2019 -0400

    Merge branch 'type-infer' of https://github.com/al3623/rippl into list-comp

commit e69b372224eead7307759f2bfbd95faf7d0b0bdd
Merge: c48ba79 4ce8d16
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Tue Apr 30 23:02:37 2019 -0400

    Merge pull request #13 from al3623/list-comp

    List comp

commit 74b01786d74a57264a652d9ba517650d9c2d0492
Author: Amanda Liu <al3623@columbia.edu>
```

```
Date:   Wed May 1 18:45:00 2019 -0400

    Comment out list clauses inference because it doesn't compile

commit 8f84060ebe7a63131e71fe1ec234ea3a65254a18
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 1 20:03:36 2019 -0400

    Fix merging bug; typo in mgu

commit ff1158fe8d658da65dcb1ccf04971a6146b830d0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 1 20:27:42 2019 -0400

    Fix inter vdecl polymorphism

commit 6444d0702e660b0f3bdcea057cc8ec06fe7f5fa0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 1 20:36:10 2019 -0400

    Create polymorphic tuple testing inference

commit 6bfb31efa763220bb84458b37367d7ea57eb3415
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 1 21:16:47 2019 -0400

    Add flags to compiler

commit 26419398360f3fc648413fa4aa8e99f7daa1e9c4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 1 21:25:29 2019 -0400

    Rename compiler executable and comment out print statements

commit 55164551e58d68a28ae1b01ac374535298db154a
Merge: 2641939 e69b372
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed May 1 21:31:06 2019 -0400

    Merge branch 'master' into type-infer

commit 9d8d2ba0de19b957e035205c994bd3371f1b3c19
Merge: e69b372 5516455
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed May 1 21:33:06 2019 -0400

    Merge pull request #14 from al3623/type-infer

    Type infer

commit 7ce063e000ed30430c468a0f76c51bbc1759c74a
```

```
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 1 21:34:37 2019 -0400

    Fix typo

commit f1fb6bba0caae2ac0e10c13db63a51817928435f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 1 21:36:11 2019 -0400

    Fix typo

commit e98cbdb46ebec06a588545d98b2fe9658b48c931
Merge: 7ce063e f1fb6bb
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 1 21:36:57 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into type-infer

commit e62e6b27374d4317b521f9fbdf0dc0d687d53c37
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 2 12:07:24 2019 -0400

    Fix error messages and try to fix linking

commit 810ed878d093683ce874746c54a251f19cde50de
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 2 12:41:54 2019 -0400

    Link static libraries with compiler so it can run in any directory

commit 4d02d297479cbed0331510a7a83dcdde1b788c71
Merge: f1fb6bb 810ed87
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Thu May 2 12:45:58 2019 -0400

    Merge pull request #15 from al3623/type-infer

    Type infer

commit e69678d7c5a684e2955672eaebc6b8b586f2a486
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 2 22:20:53 2019 -0400

    Add helper functions for listcomp inference

commit b00670d13d9dc3ef1a72c3063cef0aaccaa50597
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 2 23:23:47 2019 -0400

    Call listcomp inference function in full ti function
```

```
commit 0bb05f1f821b02b6919c21b55a3e99ad4a6ad811
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 2 23:59:39 2019 -0400

    Fix bug in list range inference

commit e25de8c9585bb50d5a26cdf03c40e18262fedd62
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 3 11:42:07 2019 -0400

    Fix merging bug for captured variables in list comp vbinds and filters

commit 3fead1c0892054f686267a7ca5399206ed8fbabf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 3 12:01:27 2019 -0400

    Find lambda Not_found opt errors

commit 5251079341a92669b0a80e3ae0aaf487871f9333
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 3 12:02:12 2019 -0400

    Fix opt to find_opt

commit 7c8a68c82a07cb7bf8e66a1487b810a924008e7d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 3 12:45:43 2019 -0400

    Account for lambda assignment to vdef and change order of vdef appending

commit db48d5a28fef7d3f44b61040e55036b8e68fd090
Merge: 4d02d29 7c8a68c
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Fri May 3 12:48:22 2019 -0400

    Merge pull request #16 from al3623/type-infer

    Type infer

commit 764b849ee73d61f487667697801e3b64215a9df1
Merge: 0953c70 db48d5a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 3 12:49:43 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into lambda-lift

commit ca6ff50462085913761b318be1866150c75d1a03
Merge: e6aec8d db48d5a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 3 12:52:16 2019 -0400
```

```
    Merge branch 'master' of https://github.com/al3623/rippl into rippl-dev

commit 1db11df37dcbbabee2f0da14e5fbac3473661186
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Fri May 3 12:57:17 2019 -0400

    Starting code for codegen of function bodies

commit 0366ca3907c17a831d78129aa79dbbd017abeb69
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 3 13:59:19 2019 -0400

    Fix listvbind list parsing and add more typable test cases

commit e165caf65d2187969c59e640f98e951e1d5f76b2
Merge: db48d5a 0366ca3
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Fri May 3 14:02:24 2019 -0400

    Merge pull request #17 from al3623/type-infer

    Fix listvbind list parsing and add more typable test cases

commit 8cb095d758e77a339786a461e65b5e4bed54f7cc
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 3 14:59:00 2019 -0400

    Delete merge conflict artifact

commit 1da4bd2f618598c1bfa811ec1dc2a31606dffb16
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 16:29:02 2019 -0400

    Fix test cases

commit 75eadb1ac41c2f9fdad5d3172916016813274281
Merge: ca6ff50 1db11df
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 16:29:14 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 62b930d0a9fbeb2a621d1e31eaf962908d6bdab0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 16:38:15 2019 -0400

    Fix ambiguous grammar with listvbinds

commit fa0be723b1ea00a1b51758aaa03d84b94a2e2b1e
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon May 6 18:44:44 2019 -0400
```

```
    Add some pattern matching for build_expr

commit dc5e23176042205313258c897a1b5da084e5bbb7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 23:02:56 2019 -0400

    Fix level of indirection in thunk

commit 136fbc2dde49f67f4e90c657670a434c21aa206c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 23:11:32 2019 -0400

    Change thunk representation in mymap example

commit 6c101b7a61518962c5ce081cf2f3c5d39e9f581d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 23:21:47 2019 -0400

    Change init thunk to take a thunk pointer rather than mallocing

commit 549577f7cb8760fe85082765cc1295a8424c9fb3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 23:22:53 2019 -0400

    Comment out second main

commit f33a1eecdaa0c9e9998c3c1543478d7d95920009
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 23:30:39 2019 -0400

    Refactor to prepare language natives

commit 55859bf0bbad066167805afcdac485917ab86731
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Mon May 6 23:36:45 2019 -0400

    Declare thunks

commit 35e03bf0d9e55bb34cf5101ecf1bd31cbcd52b1c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 23:46:41 2019 -0400

    Refactor to include Natives module

commit 1fffd29b9f12d74d36b3a9ceec5da328b9e1e264
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 6 23:48:53 2019 -0400

    Add natives files
```

```
commit 575e32ceee0879f5334421b104379f124a793a23
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 7 00:05:24 2019 -0400

    Add neq and sub

commit e9f8fbee0402f8c45172adc9424edbb73862162a
Merge: ddb8e31 62b930d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 7 00:07:12 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into type-infer

commit ddb8e314fd5f6dd6b429dac13bb9173dd364e017
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 00:07:13 2019 -0400

    Allow merge of float/int operations

commit c86e913a193ce396ee0b25b51348ad2e7a45ff1f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 00:11:34 2019 -0400

    Add space to type inference error message

commit 490d6f922d635ed95a3e90104f5c3c2d51fd739d
Merge: e9f8fbe c86e913
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 7 00:32:20 2019 -0400

    Merge branch 'type-infer' of https://github.com/al3623/rippl into type-infer

commit c509ae84a762450e7b7080bf058c064d1d66768c
Merge: 490d6f9 702af44
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 7 00:40:37 2019 -0400

    Merge branch 'type-infer' of https://github.com/al3623/rippl into type-infer

commit 702af44c17971cfc9e0bd18993743d2f97b64bf5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 00:41:21 2019 -0400

    Add lifting breaking test case

commit 03ada11e89d6b15ac4753cbcc8b6269def58b8dc
Merge: 702af44 62b930d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 00:45:11 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into type-infer
```

```
commit 4ebd87451b29267f06dfb01ae767b2873f69b7fb
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 7 00:51:02 2019 -0400

    Traverse let bindings

commit e869b59a754bb641005937f02c6329e309a1a667
Merge: 03ada11 4ebd874
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 16:56:07 2019 -0400

    Merge branch 'type-infer' of https://github.com/al3623/rippl into type-infer

commit 18af9a4f7db0098c6b0a04657c32743ac25105a3
Merge: 575e32c e869b59
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 16:56:44 2019 -0400

    Merge branch 'type-infer' of https://github.com/al3623/rippl into list-comp

commit c8e0c268d39006540690896a3984e56efc7b4da3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 17:03:04 2019 -0400

    Add sub implementation

commit 1af904ad423837e8fa3e92ad6795c790902dde8d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 17:09:32 2019 -0400

    Make names of natives consistent

commit 5f0084e001d9fa5f3c6e9ad9921a080b59825444
Merge: 75eadb1 55859bf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 17:11:31 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit ba592df02296d0dcfc7ad406f585fc8dd4e8c031
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 17:55:20 2019 -0400

    Change list natives to take thunks and add ot natives c and h files

commit 76be525f530f3be7a12ab6d81f806f37aa867ef1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 19:25:30 2019 -0400

    Fix map with new struct wrapping cat implementation
```

```
commit a963683e8193ea9e3f735b58f0c04fceb27e97c3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 20:24:24 2019 -0400

    Fix mymap examples to use apply and invoke for new cons/cat/head/etc

commit e292821502f4e2fbeffb760b43d443a143f56aa7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 20:28:42 2019 -0400

    Add another test to mymap

commit 860a0157839319200523df4c9f982e434b74f638
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 21:07:17 2019 -0400

    Working mymap examples with new list native operators

commit 444ac6bdc6600d606cca5c3d24ab938b0fe468ff
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 21:51:09 2019 -0400

    Rewrite library mymap functions to take Thunks rather than lists

commit 345fac9c2a90bb9d0355568add38a629f96a6f8a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 21:56:50 2019 -0400

    Have the make heap allocated primitives return thunks

commit 9f98b320da3e127ab26de0cf2cd580739bc95877
Merge: 5f0084e 345fac9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 21:59:00 2019 -0400

    Merge branch 'list-comp' of https://github.com/al3623/rippl into rippl-dev

commit 0ba8c987838206e394f6e307a3f22ca64cb3c48a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 22:21:38 2019 -0400

    Fix appendNode

commit 0ed22fd71178925c86e6a260bf3afebf56cf659a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 22:52:57 2019 -0400

    Revert appendNode and add getelemptr.txt

commit 0827f96c446a88a3baf34a8aaf2e508dc72a1d64
```

```
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue May 7 23:11:13 2019 -0400

    Add Hello World codegen

commit b547fdad9e1643e298454e438ee0cc44ba25e66c
Merge: 0827f96 0ed22fd
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue May 7 23:13:08 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 0d7cb1d26d00a7c2d221baab87726759641545f6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 23:24:57 2019 -0400

    Revert appendNode to not take Thunk

commit 2a9a1b51facc44e98823343e11e778f5c98ad428
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 7 23:44:01 2019 -0400

    Create thunk appendNode caller

commit 13012a3acef1b6b66bc21760a2ec8e60c0f8bde8
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 7 23:44:48 2019 -0400

    Print ints

commit 7334826c922a18cfafb3924305e5695e312713e2
Merge: 13012a3 2a9a1b5
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 7 23:45:06 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 6264ebc0de6712060fc542288b40573e6441ed07
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 00:14:33 2019 -0400

    Print bool, char, and float

commit 3e6b972613aa2961a49d5dc608da658f41b60d52
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 8 00:14:34 2019 -0400

    Create global struct_thunk llvalue

commit ed4f80fb8dbc5ed587e5b58b5629e14f9f5b8bbe
Merge: 6264ebc 3e6b972
```

```
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 00:14:42 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 4ca4b575373c09aaee923ae3c3aecb52468b38ac
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 00:25:30 2019 -0400

    Make float of type LLVM float and print a new line

commit 7d2862f2c3793e48b4fa59c391fbf253583129a4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 8 00:55:29 2019 -0400

    Fix warnings

commit ff390513701a15e22e45993d214290a6628ec5c4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 8 01:03:16 2019 -0400

    Add addf to natives files

commit 7fa10d0259c990e178cb4e908e65790770305926
Author: hlehv <hlehv@aol.com>
Date:   Wed May 8 18:41:41 2019 -0400

    Added native operator struct Thunks to natives.h

commit 39e90c8770cf6557c9c0573f03a9c7fb01c0568d
Merge: 7fa10d0 ff39051
Author: hlehv <hlehv@aol.com>
Date:   Wed May 8 18:42:44 2019 -0400

    Fixing merge conflict

commit 1022b4752d06cfedf44767b05a05a0c95abb3fc9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 8 19:25:37 2019 -0400

    Change to use global struct

commit 5d1157415be817cc886704ea77cb4b02fbe72cb8
Merge: 62b930d 1022b47
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed May 8 19:41:52 2019 -0400

    Merge pull request #18 from al3623/rippl-dev

    Rippl dev
```

```
commit 0914808ec0afd0c104a241725f246035b61c6082
Merge: 5d11574 8cb095d
Author: Amanda Liu <36548640+al3623@users.noreply.github.com>
Date:   Wed May 8 19:44:37 2019 -0400

    Merge pull request #19 from al3623/lambda-lift

    Lambda lift

commit e5511d7b2f60347bc8e3a58bf4568f24b46fe3a5
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 20:04:43 2019 -0400

    Initialize global primitive operation thunks

commit 9cdefacffe28ebf10e2aa6c1525d95108ee5e75a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 21:47:09 2019 -0400

    Declare global init add thunk

commit 783d644b70b3a022c8972db6638458c293a3f3e5
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 22:06:09 2019 -0400

    Add code gen for addition

commit de034fd69ab14766e144b121e985f7cf3fdd24e5
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 22:11:40 2019 -0400

    Remove extraneous call that was segfaulting the program lol

commit 78abb75aa33ab76185a52e175acd23f19d7a71c1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 8 22:21:07 2019 -0400

    Fix thunk alignment

commit cf0140a66242bc801b0367b4a79847b6522544ec
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 22:34:01 2019 -0400

    Match against other primitive operations

commit 356a8bca77556a805dc705030cf17002865935f0
Merge: cf0140a 78abb75
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 8 22:35:08 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev
```

```
commit c02df95ed3a70bb8ab0413447c4e374d4b25a5c8
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 8 22:39:52 2019 -0400

    Fix merge conflict

commit 00368a88ae88609bd300c49490dfe4b19a7203f1
Merge: c02df95 356a8bc
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 8 22:41:33 2019 -0400

    Resolve merge conflict again :)

commit 683c4983daa3131316932710ecaade93619ccbdf
Author: hlehv <hlehv@aol.com>
Date:   Wed May 8 22:44:21 2019 -0400

    Add ite to lib.c, lib.h, lib.ml

commit fbab70c7d1fae93216dd365f6bb0bf332ccf3cad
Merge: 683c498 00368a8
Author: hlehv <hlehv@aol.com>
Date:   Wed May 8 22:45:25 2019 -0400

    Merging

commit 987e81af3102d94f8da7b6a06b735b54bc1bf962
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 8 22:46:31 2019 -0400

    Tlength to TLen

commit 6f465b82fa683d2f37dfe908147daeabfea5cbe2
Merge: 987e81a fbab70c
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 8 22:46:52 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit a5b2f2017932ccc069f2231a2bebdf5c83c468be
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 8 23:07:10 2019 -0400

    Fix inference

commit 9b0f10221b2147de7d62b67eb2897312d931bcb9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 8 23:24:37 2019 -0400

    Get range lists in build_expr
```

```
commit 47152ea8b0c0da46a8e2f3e7abccfefc38de5c98
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 9 00:33:31 2019 -0400

    Start codegen for eval functions

commit d0649d1b68e17aecfe4dceef527961087343bec5
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 9 00:40:43 2019 -0400

    Change return value for init thunk and build lambda thunks

commit c9d3b0a075de56a98f07a29ec71cc9825a942b3c
Merge: d0649d1 47152ea
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 9 00:41:09 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 7365b224323d8f24941246a4e6c8b59808641638
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 9 17:10:29 2019 -0400

    Change type of function declarations and make global

commit 8d96c1d35ff41418b72ef70f0c1d11e33f09ae66
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 9 17:31:39 2019 -0400

    Finish build_eval_body

commit d6a9a249089f5146a12518c7165b07df7b71cf28
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 9 17:35:21 2019 -0400

    Remove comments

commit e19a59cfe4126d099f2679f16373705e379a4728
Merge: d6a9a24 8d96c1d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 9 17:35:25 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit d36455026c456fd72053cd57fef4533749f6e7fa
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 9 18:20:57 2019 -0400

    Try to fix seg fault in eval codegen
```

```
commit 194baa29bc1e2e79428198c54c22c38dc08d47b1
Merge: d364550 e19a59c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 9 18:21:05 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 8118c9c636e9ae0282dcaeef202b1cda48f9b97a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 9 18:39:17 2019 -0400

    Print another decl to debug length of param array

commit 62abadabd75ecf26ff295e8214049584cf352548
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 10 12:20:17 2019 -0400

    Fix seg fault by making declare globas declare functions for eval functions

commit 8aee25d0f5ac86dd25b419e911f9a1e2acbd8e9d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 10 16:29:01 2019 -0400

    Change error message for printing arrow types

commit d759867836aef37ca2be20383e15f09bddefa5e9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 10 16:54:43 2019 -0400

    Start build_func_body

commit ef7817a99d8057f05bfd1954d34368311a173756
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 18:39:12 2019 -0400

    Add the rest of the native operations; still need to test unary minus;
    remove natives.ml because we don't need it

commit 6503692b69cd306eca74d5a9aaad91e95e639edd
Merge: ef7817a d759867
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 18:39:28 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit b9163ee53e1aa11577e979d4350d10e790d1cb36
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 10 18:44:48 2019 -0400

    Change declare functions to define
```

```
commit 7e99ae1e173eebece440dd6720189ed697c141df
Merge: b9163ee 6503692
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 10 18:44:59 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 140ba9a5d9e81368f0866edb1f9821f99f000eb8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 10 19:18:51 2019 -0400

    Change the types of declared function args to all Thunk*

commit bbbabe7c3f988e25c42ea4a5e47135c25720c3d6
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 19:24:13 2019 -0400

    ahhhh

commit 8753551a5d87cfc4eed1850bb0a5b2f326a03e63
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 19:25:57 2019 -0400

    Readd natives.ml

commit 18fc3eb295e9e34bced800d1a22c2409fbeb7bf3
Merge: 8753551 0914808
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 19:35:06 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl into rippl-dev

commit be6d0cbc37249654c0971f98ffcbac84b741b745
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 20:20:50 2019 -0400

    Uncomment some code

commit c75414c78cf57e4fce950ee65bf8d4f98dbf60c9
Merge: be6d0cb 140ba9a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 20:21:54 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit c85f0bc0b1581177539a0456dfc73fcaf8e5a613
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 21:01:31 2019 -0400

    Fix C implementation of pow and powf
```

```
commit 121d48ae1e98d0b7824422df89508104aadb5154
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 10 21:13:44 2019 -0400

    Unmangle function declarations

commit f28f0ac09bea329f05a8fa6f7daaef72b976467e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Fri May 10 21:44:51 2019 -0400

    Add terminators to eval and func

commit 37a63a37a99b7b4a1a26364f95f843b9920d211c
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 21:47:35 2019 -0400

    Throw away lambdas to isolate lbody

commit cc079dc7070a4ca927a333381855ffaa411033c0
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Fri May 10 22:02:41 2019 -0400

    Rearrange definitions

commit 2c00439100c634c101fa651ee4dba81991fc8424
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Fri May 10 22:18:04 2019 -0400

    Add build_eval_func_body implementation

commit 6368ec32edd6a38c14ec3bf411f1ae590a1b9af4
Merge: 2c00439 cc079dc
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Fri May 10 22:24:12 2019 -0400

    Fix merge conflict

commit 9706a9f416466beddb0ba9be9790aedd0c6173df
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Fri May 10 23:06:50 2019 -0400

    build_eval_func_body works

commit cb70bc4e03d8345254621dac3be2f235f6a082ce
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 00:15:32 2019 -0400

    Finish generating function bodies

commit 7f9d7d2713a55edea7a40c4976ab95afe5bacda7
Author: Hans Montero <hjm2133@columbia.edu>
```

```
Date:   Sat May 11 16:24:15 2019 -0400

    Look up for func_init_lambda instead of func, tweak thunk.c
    to allow for full currying

commit 118ebd9cb2b9d40db5f7dbf0a9f937ee782edccf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 18:38:34 2019 -0400

    Remove old debug print statements

commit 1df6805f37f0572adb11aae4356ea7c421183b2d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 18:51:38 2019 -0400

    Change makeRange to take Thunk *

commit 7811ec2a08d7ee6d0d43bd68e27d49da8760d03e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 18:58:22 2019 -0400

    Change makeMaybe and makeTuple to take Thunk *

commit 5b6ead6ae3e9c916d761a04f1228ede81501a475
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 20:25:28 2019 -0400

    Add tuple, just, and maybe constructors to build_expr

commit ee610dbde20297b129bdb3bae0a1b58d54920a3d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 20:26:29 2019 -0400

    Remove stupid test cases

commit dd28b4f8ff7c66bec388437e74ab37179deb4640
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 20:42:46 2019 -0400

    Generalize printing of lists

commit 19cf945bacfa4bbabf11fe644e36370046ab5985
Merge: dd28b4f ee610db
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 20:43:27 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 28f1a71b5a38e8589b2ecc849494d3b97aa42cbd
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 20:49:03 2019 -0400
```

```
    Remove a comment

commit 3b5c9e1a368db27b18c086faf82de17cd6e3a0d2
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 20:49:46 2019 -0400

    Add printing for maybes and tuples

commit 7eee2135029678b548e4be24fddcdaef48ecaafd
Merge: 3b5c9e1 19cf945
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 20:49:52 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 4afae93d1a6b8a2f042cb7c6cb54807666f2bbd7
Merge: 28f1a71 7eee213
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 20:49:10 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit e596a7c32528c76e0fab935509e2cacc90930ba0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 20:52:37 2019 -0400

    Remove redef of print functions

commit f16b08fed1ab59c98c7ed79b3066080262f98d6d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 21:02:49 2019 -0400

    Make call to making tuples and maybes type consistent

commit 529aebded2cf812a157a7cc80cf6cb6206e57dd5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 21:32:13 2019 -0400

    Update type pretty printer for arrow types

commit 7dc48ae81936f1759c2e531ff9034ad1a5cb454b
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 21:33:50 2019 -0400

    Delete debug statements and comments

commit 55caac29a0fbff0894ebb0cfc762f9f3260000c0
Merge: 7dc48ae 529aebd
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 21:33:56 2019 -0400
```

```
    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 9afc0994600948dae7ce9879a5876c864b9ea807
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 22:10:48 2019 -0400

    Adding some stuff

commit f94bffb0af12eab827366f29a6ff6b46b85e7ba4
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 22:18:44 2019 -0400

    Rename stuff to avoid conflicts

commit b1dd2f60d9afec52348778e224f268e149d0a84b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 22:23:37 2019 -0400

    Fix composeSubst

commit 354f3a3ee7c828242feb2d179e6eef25e68f5b16
Merge: b1dd2f6 f94bffb
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 22:23:48 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 45d067d7c59c1c78ec2287115d5619c02be20eb8
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sat May 11 22:40:39 2019 -0400

    Implement boolean thunks

commit 3c5f4c2b23c3731cb2ae4d426d85f9693ffc89b6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 22:48:19 2019 -0400

    Fix list access to not call invoke on thunk value

commit df1ae164305b2f937c6b82dcc49b0e7b9c3626a2
Merge: 3c5f4c2 45d067d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 22:49:52 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit e78afb237fc331295a28e1836226e593feb822c5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 23:06:14 2019 -0400
```

```
    Add multivariable filter example

commit a8f9a1095ad14e418d4042e43952172a6b1dcf51
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sat May 11 23:48:47 2019 -0400

    Fix range generation to not depend on literals

commit d63d5d48ff40bd4564fea999cd531d54add4d0a5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 01:03:06 2019 -0400

    Add int_to_float to pre-codegen parts of compiler

commit cefd64fdc51436432611f3bd8050b9045d24ff31
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 01:25:36 2019 -0400

    Start debugging int_to_float thunk error

commit 0e14a495fd96b2c6fb8a1f6577ae0347b25790fb
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 01:29:54 2019 -0400

    Fix int_to_float thunk error

commit bf1e22f564a48498c67d552eaf1aec8518aa106c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 01:52:28 2019 -0400

    Add wildcard case to catch listcomps

commit 90988927f40cce950008e7617f3c761261021ee0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 14:34:56 2019 -0400

    Fix bug with printing lists that start polymorphic

commit 49e0e24960484b43298df516f89c7a2b1d7fc45e
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Sun May 12 16:29:03 2019 -0400

    Codegen for List comprehensions ^_^

commit ed1ace13889884bfc7a731d7f1257bc8a6e39ecd
Merge: 49e0e24 9098892
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Sun May 12 16:29:10 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev
```

```
commit 04574d9dc423812176ab200cb378e9099a15f243
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Sun May 12 17:25:25 2019 -0400

    Add listcomp test

commit a247da7e631375fdd34248dc6ebd4d4374bf8205
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 17:43:48 2019 -0400

    Add broken directory to keep track of bugs

commit e1c7dff0f7480e768233c13a089c33ec886c3934
Merge: a247da7 04574d9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 17:43:59 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit d8f0e6417730e0e7b093e59730fa8ace41c4be13
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun May 12 21:59:54 2019 -0400

    Add tuple support to lambda lifting, slightly tweak tupleSquare

commit 75d641472791e1fc18fb4adf1b5436b37809197f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 22:02:09 2019 -0400

    Add tuple ast printing

commit 43b166e045faab2da656dbd8bc80827b6d8cc058
Merge: 75d6414 d8f0e64
Author: Amanda Liu <al3623@columbia.edu>
Date:   Sun May 12 22:02:20 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit db6712ed36e3033ba05cdcd5ed0e5e0ef7d6e555
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Sun May 12 23:28:24 2019 -0400

    Support list comps with one listvbind

commit 9868e4831bc68c6c93677f0fe0d38ac5fc5f6f82
Merge: 43b166e db6712e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 13 00:04:44 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev
```

```
commit d85e4f2a62a7b39e795007da110028ef52116734
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon May 13 00:09:52 2019 -0400

    Cleaned up compiler warnings

commit 743ec3368ff8803c5d2ab6d9d6e7accc009fad6d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon May 13 14:58:14 2019 -0400

    Support Just type in lambda lifting

commit a41b8e8ab6408618c0f04000e95a0430e91c2853
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon May 13 15:05:21 2019 -0400

    Throw proper Failure when non-main vdef isn't an arrow type

commit f0e00a40db2104b674f865f7b56031ca98711cb1
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon May 13 15:15:47 2019 -0400

    Throw Failure if no main program

commit 7411e1fa0fd2d7b976d25ee8ea86d2121ba866fe
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon May 13 15:43:16 2019 -0400

    Properly detect lambdas inside of lists and pretty print list contents

commit 1f62a9587767e0a8f307faa642de8f6773b42ab6
Merge: 9868e48 7411e1f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 13 15:48:13 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 11b307af974ffb6449ca61646ae6880f2d9f88b2
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon May 13 16:14:44 2019 -0400

    Make compiler flags just do their specific function (don't also do codegen)

commit 9c7ff103c79bbb7ffe96f9cc2f4352e754fe5f2c
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Mon May 13 16:30:42 2019 -0400

    Revert flag changes

commit e43b3b5d24075cbb1ad3e1286a64136a368f4da8
Author: Amanda Liu <al3623@columbia.edu>
```

```
Date:   Mon May 13 23:31:56 2019 -0400

    Update tests

commit 7c3b82b1abddf6312ad4bc9ec0ea30d255010056
Merge: e43b3b5 9c7ff10
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 13 23:32:04 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 5bc6158973d74b66fa5050405d9855b3b535d98e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 13 23:39:24 2019 -0400

    Change type of filter to be consistent

commit 66565fb47057ad31af1a862ef4ada64e9792168e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Mon May 13 23:41:11 2019 -0400

    Fix extra pattern match warning

commit 6a05f3c67b705f8b924f0d513ba28375934882ca
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 16:56:30 2019 -0400

    Add test case for mandelbrotRange (currently working)

commit 8baff3f9368184a4902df703c754e9eb0c6dd4d8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 17:26:44 2019 -0400

    Test cartesian plane for new mandelbrot implementation

commit 1564560ab3e7b4b8249aeb2408a4d51cf43950be
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 17:53:46 2019 -0400

    Update some test cases

commit e1c2e3f54c94011c178e8897ef4ea1bf33eaaaf0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 20:03:34 2019 -0400

    Add checking between inferred types and annotated types

commit 2c636c07edaf8ffdf70b1b3ee2e126b551e2df12
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 20:13:04 2019 -0400
```

```
    Fix parsing of arrow type annotations

commit fd811bbdb20c5223e26ae3141b7c5977bb4cdcaf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 20:39:13 2019 -0400

    Add scary mandelbrot that broke clac

commit c45574af70977f71b8ffaaf183858ce8c8326162
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 14 21:55:40 2019 -0400

    Unbreak Mandelbrot because it always worked just not on baby CLAC

commit d2e3337179b8109aef7e5bb41766a61520c15b38
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Tue May 14 22:02:27 2019 -0400

    Add fibonacci

commit 2a073e3f39badcdf2563f8ba4ce7ed47517a8421
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 14 22:19:52 2019 -0400

    Remove print_endline call

commit 4e7e855f3a03fec8362ebdc58a7882dd8262f3f0
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 14 23:08:51 2019 -0400

    Mandate type annotation with main

commit fbc97f2e3f7da48bcfa23ff6abfc962184999658
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 23:09:13 2019 -0400

    Add type depth calculation and array allocation in main codegen

commit dbf4df58f5c5440fc60faf6d6e4da7dccd74f7ea
Merge: fbc97f2 4e7e855
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 23:09:22 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit df3556fceed6e145b419d3f92506c5aa69fc746c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 23:33:29 2019 -0400

    Create heap representation of nested types for main
```

```
commit d213dd6751f7fce645f3c2f74f1f024ce80921ed
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 23:39:29 2019 -0400

    Fix mandelbrot type annotations

commit a67e042fee69447e694c62d37872f96ca8b8592b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 23:44:58 2019 -0400

    Change function header for printAnyThunk

commit 1f27cdcb4a94c67087fd9bdc9095d9c6f4af39b2
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 23:52:59 2019 -0400

    Change other type signature

commit 88dc299ea695a44e0b2e88849506c6238a3d0bdb
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 14 23:53:06 2019 -0400

    Make print_any_thunk take a type heap

commit 67bed09564c663a5abdfac5d01db16c64121e5d4
Merge: 88dc299 a67e042
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 14 23:53:12 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 654484f9a075e1204caec63265ff2952a7a67fc4
Merge: 1f27cdc 67bed09
Author: Amanda Liu <al3623@columbia.edu>
Date:   Tue May 14 23:53:28 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 72b3e31461b8f8fdca22ff90ade6be3437811706
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 14 23:57:57 2019 -0400

    Delete printAny (replaced with printAnyThunk)

commit 375206b09331ac0e75240b7c148af0441eca9d4c
Author: hlehv <hlehv@aol.com>
Date:   Tue May 14 23:58:03 2019 -0400

    Add is_none and from_just

commit da9b825d1f50a2725b4de30b8b43a5c5d1100fd5
```

```
Merge: 72b3e31 654484f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Tue May 14 23:58:04 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 74e9b6cd8eaa94193c18cfe0019754db18a019f2
Merge: 375206b 654484f
Author: hlehv <hlehv@aol.com>
Date:   Tue May 14 23:58:13 2019 -0400

    Merge branch 'rippl-dev' of github.com:al3623/rippl into rippl-dev

commit 80cf152b830d143572f007b49b0c19f76df7506a
Merge: 74e9b6c da9b825
Author: hlehv <hlehv@aol.com>
Date:   Tue May 14 23:58:24 2019 -0400

    Merge branch 'rippl-dev' of github.com:al3623/rippl into rippl-dev

commit b8e4f61eb6297c80afa8034ee19ee1f6faadca15
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 00:03:34 2019 -0400

    Fix printing to accomodate new type signature

commit a6dad695b36b232b16fb475ab1d959a5b441baaf
Merge: b8e4f61 80cf152
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 00:03:45 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 19fc4e8ead9b96b2ef77a52fd2543e50c4d62c51
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 00:12:53 2019 -0400

    Fix exponent error

commit 6801dd16791b075cde05f01b4401bee433d95e6d
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 01:18:01 2019 -0400

    Fix thunk application and modify a few test programs

commit e28ca59b3b190c1005d63894b88d5f1f153d7a87
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 01:18:15 2019 -0400

    Move tuple_square out of broken
```

```
commit 3064e07fc16031ef253b29c7ea6d46521c0ad71d
Merge: e28ca59 6801dd1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 01:18:21 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit af9c884fd58de6a38d625442189e16622edb0376
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 01:26:17 2019 -0400

    Add old version of mandelbrot with list comps

commit ffac32cfd0abe2e3493807e7995eb728bdb00560
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 01:26:27 2019 -0400

    Add some more test programs

commit 99de13c93f5138c440cffd261326f38ca401f352
Merge: ffac32c af9c884
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 01:26:31 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 64a2b06e14afbe661ec6d10fdce3e38b85f5fdc6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 01:56:14 2019 -0400

    Fix list comps for mandelbrot

commit 6ffffa6181f99785044b46ce26759853b80a3eca
Merge: 64a2b06 99de13c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 01:56:26 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit bfd971af7dd255863b34ea803d0a86801b577384
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 01:58:22 2019 -0400

    Remove debug statements

commit 01891627be5155ed7f098d5bdac21f3b4c160a17
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 02:04:44 2019 -0400

    Remove cartesian example
```

```
commit 44215f41b0c12065b9b97b2d947e2ab7751e61cc
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 02:04:48 2019 -0400

    Type annotate a few test programs

commit b94a987cf819e81e344b415859e7f73e6935bdec
Merge: 44215f4 0189162
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 02:04:56 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 14310e78007f14a3ee12139f874cbfa5d27389b5
Merge: 0914808 b94a987
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 02:06:12 2019 -0400

    Merge pull request #20 from al3623/rippl-dev

    rippl-dev

commit 16e12d7ad39f404a9f1a6dda703a55ce1608a01f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 10:43:33 2019 -0400

    Add julia set example

commit eb83974fb6d3cabaac3c0cb86eb32e435fbf98b3
Merge: 16e12d7 b94a987
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 10:43:49 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit b527543456526ee253aa81c72168acb9222f36ea
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 10:44:04 2019 -0400

    Add fun_tuple program

commit dc8eee4258d936aa64119145f7234f9dc29775f1
Merge: b527543 eb83974
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 10:44:07 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 70bf57d1ad5d9dfa7be9fddebe07e22de96ea887
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 10:48:02 2019 -0400
```

```
    Modified listcomp test program

commit c34e5858b622e310e5dc071e47c9aa16cf5df642
Merge: 70bf57d dc8eee4
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 10:51:02 2019 -0400

    Modified listcomp.rpl

commit d6eb7ba26494e13dcf558aabf7bf55f11c077791
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 10:59:13 2019 -0400

    Fix application error in julia and reveal lifting error

commit 9bc77d98b504b5c5281464a1baaf2f1b77425e4e
Merge: d6eb7ba c34e585
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 10:59:25 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit c415b3fe9b1138c62340db2ddde13d10cd7d68fc
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 11:05:09 2019 -0400

    Remove debug print statements

commit 5ff3e5088f752f9b48e212ee1400a6cfe8e09912
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 11:16:45 2019 -0400

    Compiler will not compile if flag is given

commit da79a2dd5539bedc522a171066e55f04549d6f76
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 11:17:07 2019 -0400

    Iterate through ITEs for closure vars

commit 9e997567eb773a47c3042df9088ef995aeebc346
Merge: da79a2d c415b3f
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 11:17:09 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 3c35ae16a80298cf6e67424413d1e27b3ac6c452
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 11:38:07 2019 -0400
```

```
    Got rid of the pattern matching warning in inference

commit e3a725a1f883e3583311972753b790c0ede9f048
Merge: 3c35ae1 c415b3f
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 11:38:14 2019 -0400

    Merge branch 'rippl-dev' of github.com:al3623/rippl into rippl-dev

commit e68a41c32718f476096f277e17c7e9f60de02c1a
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 12:07:09 2019 -0400

    Removed unneeded code in lib.h and structs.ml

commit 491cbb396c384f9aafde626f3191ce3929786282
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 12:23:19 2019 -0400

    Added hollis.rpl

commit 43caca76bd7ddaacd3c12c183d8bed20355e47d7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 12:24:21 2019 -0400

    Fix issue with filter

commit 2217b1c2901ef4cd4db84f1bd0062003120ee538
Merge: 43caca7 491cbb3
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 12:24:36 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 962004ae2065b61b7e2335b223850d3a91011c81
Merge: 9e99756 2217b1c
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 12:26:47 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 7ded25c26e7f2a998f3d3252d9e84c1a813c01c0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 13:29:45 2019 -0400

    Add brokenITE example

commit d2e432f98a4b2fefdb7d014a43ab0e3167f1ea79
Merge: 962004a 7ded25c
Author: Hans Montero <hjm2133@columbia.edu>
```

```
Date:   Wed May 15 13:30:04 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit ed214145e9d6428c4b356cbb41170b6f36a1a7ee
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 13:40:47 2019 -0400

    Remove print statements

commit 1bfa47a5a33e77401e9ce2dbad91318c1ac77a67
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 13:55:36 2019 -0400

    wrapped ITE in a thunk

commit 16ee77b0e3ff81f892a97907b790e8d54b960165
Merge: 1bfa47a ed21414
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 13:55:42 2019 -0400

    Merge branch 'rippl-dev' of github.com:al3623/rippl into rippl-dev

commit 960e810516e694edb0809d0dbd9ad8d74877a6db
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 13:58:23 2019 -0400

    Clean up some comments

commit d431f37a013d63bf235c72684fd32e0b45237f8b
Merge: 960e810 16ee77b
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 13:58:26 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 176d8c9af12b586a9232658418f49ccb9f1d7e12
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 14:33:29 2019 -0400

    Change julia

commit 2b3afbc8aab3cdecb82badf649c6fde1445da14d
Merge: 176d8c9 d431f37
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 14:34:28 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit f2ec5e7f7fe69199bccff3782ac7c149e836696c
Author: hlehv <hlehv@aol.com>
```

```
Date:   Wed May 15 14:43:55 2019 -0400

    Modified thunks for ite

commit 8fb80cf25d50df3d8554189de3d2a1a76ff16877
Merge: f2ec5e7 2b3afbc
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 14:44:19 2019 -0400

    Merge branch 'rippl-dev' of github.com:al3623/rippl into rippl-dev

commit 6cbedf2ac53a62ed604a7282f2949eec63170b1e
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 14:52:41 2019 -0400

    fixed num args error

commit 4ce67538cbb1d75834689a42305b76e44df279c6
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 15:01:03 2019 -0400

    modified struct in structs.ml

commit 097b71c6ee625c3d3a8cb9f9754a8822cba66a88
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 15 15:09:41 2019 -0400

    Add mymap.rpl test

commit 0905477d3b36a6fe70a9b373a46d20fcd3f360aa
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 15:13:04 2019 -0400

    Make main in julia call julia

commit 4d587c363e0c3bce50b779b61216b6d2e3c5e82c
Merge: 0905477 097b71c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 15:13:22 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 27e09ed38df8fa74f2430512a746890cb87d2306
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 15:17:40 2019 -0400

    Set alignment for all init thunks and remove some print statements

commit 95d7695f91ef6414c5b9d0fa4c67394bd84fc75f
Merge: 27e09ed 4d587c3
Author: Hans Montero <hjm2133@columbia.edu>
```

```
Date:   Wed May 15 15:17:47 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit c9bf3a893547283f1923f42dca75e64cd2e2bd32
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 15:19:15 2019 -0400

    Remove a stringset iter

commit 5c3af6fd07944652c5271e63fd3dd46a6f9f9a00
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 15:52:46 2019 -0400

    Add debug test case to print Julia set's plane

commit 75e90136d68f8668c4b967cb5e9a0184a3f031d9
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 16:28:09 2019 -0400

    Create demo directory and subdirectories

commit 4290bac3038520ec25265ab3e7b6b0dfe2b03125
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 16:56:48 2019 -0400

    Don't remove wrapping fun from main

commit 67799dbb1baf64c7e24981264bfba8acfd486d55
Merge: 4290bac 75e9013
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 16:56:56 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit a6adaee6044ed2c188bf2389d04b7d626b884e88
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 17:49:40 2019 -0400

    Remove old syntax example

commit 03164bf4aa38a4033a827c7714f30dcc386180d5
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 15 17:49:55 2019 -0400

    Add onehotrippl.rpl

commit 7191edb669837cf4122c7b88102cb9c38ca46fbf
Merge: a6adaee 67799db
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 17:49:56 2019 -0400
```

```
    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 58ce8a9fd2a510e36e9b973b92f2bfaeb7ba589c
Merge: 7191edb 03164bf
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 17:50:22 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit eb235aba5223bddecd2cf520953b01c48cc6d4f2
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 19:06:56 2019 -0400

    Fix type depth calculation

commit da98559548b200c9558dba80881f89e31fa83ab4
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 19:07:21 2019 -0400

    Remove print statement

commit 6188b14816f1fdabb7cf93d999ea196467295bb6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 19:13:39 2019 -0400

    Add support for maybe annotations

commit 67db0d8e6ac25833251fd8cf8afb8f67087385c1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 19:19:26 2019 -0400

    Make type pretty print lowercase

commit 8fb5ab827c205dd06de44bb2f0a0944aaa0aad84
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 19:28:58 2019 -0400

    Add operators to pretty printer

commit 230c67e4c1f610e8978316cf14fade14dd954f44
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 19:31:43 2019 -0400

    Remove infinite lists

commit b6add50eeed048eb82ee5c705a489f2fcc60ce0e
Merge: 230c67e 8fb5ab8
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 19:35:45 2019 -0400
```

```
    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 8dcf1d2e73455079c2a68fe821332c705a9392ca
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 15 19:41:36 2019 -0400

    Add gcd.rpl

commit 6a9660e702d9989698403af8b22d9003fc2c4780
Merge: 8dcf1d2 b6add50
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 15 19:41:55 2019 -0400

    :qMerge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit da9164de8159033c5d742633ae7fa1964e4b13be
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:00:05 2019 -0400

    Add inference demos

commit f521fd015347336cc7cafefffb045c29e5399deb
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 15 20:01:38 2019 -0400

    Change onehotrippl.rpl to use tuple

commit f9c9633276c9f088167efac8cc02825c593fa7c1
Merge: f521fd0 da9164d
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 15 20:01:53 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit ab4e7c105d5872c0271843f28d95f27902435dad
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 20:20:05 2019 -0400

    Print annots with -l flag too

commit 7a14d58a72b8a0b060e7ab46cb03f7e3df20d303
Merge: ab4e7c1 f9c9633
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 20:20:14 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 3ce2a21eabf66103dcfdc21c4b56300c9ab0f080
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:27:41 2019 -0400
```

```
    Add more tests for demo

commit 199b43bd5e0abc33783096a96b851e23d8069709
Merge: 3ce2a21 7a14d58
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:28:02 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit f2bbc38e23d22cb9e56e42b915ccca2344c65b84
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:29:20 2019 -0400

    Add onehotrippl to type inference examples

commit 056aab667999b7d2f96e8b983fedce69c991295a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:40:42 2019 -0400

    Cut down inference examples

commit b478c89dc3e5b2f48bca6c76f58aa1404ddfa781
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 20:44:17 2019 -0400

    Add lazy.rpl to demos

commit 2e984c7c8ce8a8b28eee500c42f4dd6aca9ff2b7
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:45:19 2019 -0400

    Add plthw1 again

commit 3da0a920fe7fc1f88d3407f479e9f1eb6a4aa73b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:50:47 2019 -0400

    Finalize demo programs

commit df9f14a55306390a4822414554e0fe1b729bd16a
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:52:07 2019 -0400

    Change arrow example

commit 2eb6a07bec8f25f69a8310942648ede329879e85
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Wed May 15 20:53:26 2019 -0400

    Remove demo/plthw1.rpl
```

```
commit 58ec350d6e6137627d95a50ae9d4fa6d5992c99e
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:53:43 2019 -0400

    Remove other programs

commit 8c1019cecf486cbe57809633d48b9ef21a289d9f
Merge: 58ec350 2eb6a07
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:53:53 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit 8fed29abf951f54f4382c06b60c8f99021f655da
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 20:55:42 2019 -0400

    Comment ronghui-nl.rpl

commit 6777c1fbfab11c2e9dc02b0da682aa09e9f951b1
Merge: 8fed29a 8c1019c
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 20:55:55 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev

commit c4ffad0537e43186176171dd004adb48ce7ece2c
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 20:56:13 2019 -0400

    Added lazy2, lazy and strict to demos

commit 44620c1f3ca314b25cb4803e844af46e149ab5ce
Merge: c4ffad0 6777c1f
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 20:56:32 2019 -0400

    Merge branch 'rippl-dev' of github.com:al3623/rippl into rippl-dev

commit 40ecddc2d5bcbb163819832fceca5d2c06fc9e53
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:57:09 2019 -0400

    Add lift example

commit b28c1fbeeb0078dd381e5b82db1b011d46d5e5fc
Merge: 40ecddc 44620c1
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 20:57:19 2019 -0400

    Merge branch 'rippl-dev' of https://github.com/al3623/rippl into rippl-dev
```

```
commit 00242b70293a4e50d43c777aac092ea0ccef59a6
Merge: b28c1fb 5ff3e50
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 21:00:47 2019 -0400

    Merge branch 'master' into rippl-dev

commit e6e6fc5e04ed8e22496814211f15bbaa9195360d
Merge: 5ff3e50 00242b7
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 21:01:26 2019 -0400

    Merge pull request #21 from al3623/rippl-dev

    Rippl dev

commit eff9a17defaae3120f1eb011477c45e3d33242a6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 21:56:05 2019 -0400

    Restructure inference testing directory

commit f3a29b34a74dca43eefb1829dc504c3310b612f0
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 22:00:51 2019 -0400

    Remove lexing test directory

commit 8fa2c081b20cd1bf9a090755f69faf280e1e5538
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 22:01:56 2019 -0400

    Added just test program

commit 4968f810e58a377ea4ccda9a42e4892254634989
Merge: 8fa2c08 f3a29b3
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 22:02:02 2019 -0400

    Merge branch 'master' of github.com:al3623/rippl

commit 4c6f65c5ba7ec85cf4c64ceff1df704591e271d7
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 22:03:39 2019 -0400

    Added partial application test

commit bf44958444b5185daceb58eb2d790dbb23f992cf
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 22:04:28 2019 -0400
```

```
    Added tuple test program

commit 78ab075c348315a27bc7fb51e5f75cdc8897fffe
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 22:12:04 2019 -0400

    Add test script

commit 2c545b0b1ecb5bc4b0a9749fc0c29fbe97bea3a8
Merge: 78ab075 bf44958
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 22:12:19 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 1a1449bda735d03b794e9c9828437e2d678d8d5b
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 22:15:00 2019 -0400

    Make test script clean directory after

commit b2ce9b3f238b961dc1259e017243960ac7c4b844
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 22:20:18 2019 -0400

    Slightly modify couple of test programs

commit d63eee3f85b1f71514ce427ce9c26f51652002bf
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 22:23:28 2019 -0400

    Add lambda lifting demos

commit 75bd4f3f03a346c764f937c1e79799039bb2cd7a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 22:24:04 2019 -0400

    Untrack byte files

commit 9b963608c79596eeea6cb174f6cd1bf2da79a809
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 22:25:44 2019 -0400

    Add color highlighting for test suite

commit 073b6e0b93c8875fd158a6b0667feccae56e07a6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 22:29:48 2019 -0400

    Add highlighting for inference tests
```

```
commit b09ee589aa6642d701ecf8177c1ab0a18a5fa89f
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 22:30:30 2019 -0400

    Remove parser directory

commit bccc6fe0f1ba404400f64692021aabdb1a132ebc
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 22:40:06 2019 -0400

    Annotate hello

commit d93f01bdc537a21c1451bedb9432cf67f2b58bba
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 22:42:50 2019 -0400

    Tweak a couple more programs, change precedence of bool ops

commit 4d87c410147912958d4ac73a965ccf6a23c1cb65
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 23:09:21 2019 -0400

    Prevent double traversal of list comps

commit 30d05616e732bf130a24989e4ffde820fe54a5ae
Author: hlehv <hlehv@aol.com>
Date:   Wed May 15 23:17:02 2019 -0400

    Changed typo in pretty_type_print

commit 89736ef612f19a5905b09811bd6fddc648c7421a
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 23:21:38 2019 -0400

    Add cl-lift tests, clean up

commit 53fb7c2f77e043e3f00e0f4441591122dbad81e5
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 23:23:29 2019 -0400

    Fix inference for polymorphic lists in vbinds

commit 4d4e5b78d159208b93cc62116ead1f0cab5366b8
Merge: 53fb7c2 30d0561
Author: Amanda Liu <al3623@columbia.edu>
Date:   Wed May 15 23:23:40 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 40972919927c730ef4c0acf34de38e355672e7a2
```

```
Merge: 4d4e5b7 89736ef
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Wed May 15 23:28:24 2019 -0400

    Merge pull request #22 from al3623/rippl-dev

    lambda lifting changes xD

commit 473b5f9f11ddd1930892f055a0b54235eec6915d
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 00:00:07 2019 -0400

    Fix whitespace

commit 53d12a4edc2edc74142d27baa136d720a49bab73
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 00:01:08 2019 -0400

    Add polymorphic id example

commit a6c32d314ae30e341d5bfb2455f26416208e1538
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 00:18:39 2019 -0400

    Change shell script to zsh and add some spaces

commit bf5ecf5f9b14ba353af83ebff407fc9ccb28459a
Merge: a6c32d3 53d12a4
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 00:19:14 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 86e458594dd627e38ff46f51229151798effeb63
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 00:53:08 2019 -0400

    Fix ordering of vdefs lifted

commit 7c42934ecb72f8e14954c08565f760bb1ffa3521
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 11:27:48 2019 -0400

    Delete TODO.md

commit e10b41cea06734d3e214454d7b8f704dc4b0b415
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 11:28:27 2019 -0400

    Update README.md
```

```
commit 6fdabca1fe78edba953a2d51a9ee96da107281d8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 11:49:07 2019 -0400

    Make mersenne actually calculate mersenne primes

commit a1bd3819987801c3d2727f54d0fb8d242d89268a
Merge: 6fdabca e10b41c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 11:49:31 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit c01415a84315d9eab5fe1b7fc6aba07401152eae
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 11:57:38 2019 -0400

    Beef up arrow_list

commit b9d81e6375cef9a6366fe913935cd580e80706b4
Author: hlehv <hlehv@aol.com>
Date:   Thu May 16 12:01:37 2019 -0400

    Changed strict.rpl

commit 50bade23ca60ec3a833f72fe4e4d0896c3b69509
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 12:01:53 2019 -0400

    Clean up demos

commit acf9e4d7cf7a216a1d0d586131e8aa96d23c950b
Merge: b9d81e6 50bade2
Author: hlehv <hlehv@aol.com>
Date:   Thu May 16 12:02:19 2019 -0400

    Merge branch 'master' of github.com:al3623/rippl

commit 18145cc0917b98104e9ee39a577bc8887de88ad7
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 12:05:51 2019 -0400

    Add one parenthesis

commit 0451f839457c1d80aac0be7572f7be53d70f1f18
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 12:08:28 2019 -0400

    Move mersennne

commit 2cdebe6d6ed35657ca0b3d3f263c369d12777824
```

```
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 12:09:11 2019 -0400

    Add test driver for lifting

commit 673d59cea50d5c941fd267ce2b69e894115e6758
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 12:10:32 2019 -0400

    Beef up arrow_list

commit 208172c802538ce4ce4d6d0d8fe1661a16979f6a
Merge: 673d59c 2cdebe6
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 12:10:44 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 230a0fbe3b584b22f411d98f6d1f89a4b3d04e57
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 12:11:36 2019 -0400

    Remove onehotrippl

commit 6f8063920abc3493960719c09a39a923a34d79f5
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu May 16 12:12:13 2019 -0400

    Remove listcomp.rpl from thunk demo

commit b65cd9ba99e80ba5297533cf7c73b0b2f7600f7e
Merge: 6f80639 230a0fb
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu May 16 12:12:21 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit d301489b1a453ae76f614cd39e70e3f56865b6cc
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 12:14:16 2019 -0400

    Remove some parenthesis from maybe printing

commit eb082b7ee423ee3add2cf4fe3a4442e80ab236f2
Merge: d301489 b65cd9b
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 12:14:24 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 55729ea8f1adc47033d33f98502b1919e29bb103
```

```
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 12:22:37 2019 -0400

    Channge resolution of mandelbrot

commit c4523565ca074ca4a29aba8200dbc15be39300bd
Author: hlehv <hlehv@aol.com>
Date:   Thu May 16 13:31:38 2019 -0400

    Switched names of lazy and lazy2

commit 9cfa21863c45850c39fe6f8b83eeffb3a61f1b33
Merge: c452356 55729ea
Author: hlehv <hlehv@aol.com>
Date:   Thu May 16 13:31:44 2019 -0400

    Merge branch 'master' of github.com:al3623/rippl

commit ae0ad9f767f6a3482d30b073c9ca9912b6dfc3f8
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 13:32:29 2019 -0400

    Remove variable shadowing in arrow_list

commit 4ff9b1127fefe3ef9eb914271d2f2edcb2590780
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 13:32:39 2019 -0400

    Update mandelbrot demo in demos

commit 86e7a745a57aeea1a24d7bc1c653b85ced767f0c
Merge: 4ff9b11 9cfa218
Author: Hans Montero <hjm2133@columbia.edu>
Date:   Thu May 16 13:32:44 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 11e57615c02df7a4eff194750b0f7fff22dfa37c
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 13:33:53 2019 -0400

    Remove variable reuse in mersenne

commit 04241b2b84a5c8286a29d2f90756cb63261d2d02
Merge: 11e5761 86e7a74
Author: Amanda Liu <al3623@columbia.edu>
Date:   Thu May 16 13:34:05 2019 -0400

    Merge branch 'master' of https://github.com/al3623/rippl

commit 9a7ad8771aacb1c7facfa07c96c1d5ad78181f0b
```

```
Author: Da Hua Chen <dc2802@columbia.edu>
Date:   Thu May 16 14:48:20 2019 -0400

    Fix onehotrippl to match slide
```

5 Architectural Design

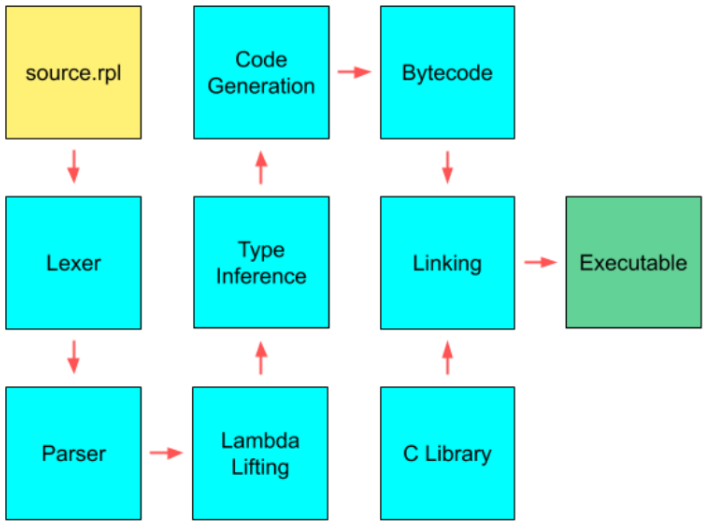

Figure 1: Rippl compiler architecture

We start with source.rpl. rpl is the extension for Rippl files. This file will undergo lexical analysis and then semantic analysis. The compiler then performs two major semantic transformations on the abstract syntax tree produced by semantic analysis: lambda lifting and type inference. Bytecode for the program is then generated which is linked with binary files from the c library to produce an executable.

5.1 Components implemented by

Lexer: Amanda Liu, Hans Montero, Hollis Lehv, Da Hua Chen
Parser: Amanda Liu, Hans Montero, Hollis Lehv, Da Hua Chen
Lambda Lifting: Hans Montero
Type Inference: Amanda Liu, Hollis Lehv
Code Generation: Amanda Liu, Da Hua Chen, Hans Montero
C Library: Amanda Liu, Hans Montero, Da Hua Chen, Hollis Lehv

6 Testing Plan

6.1 Representative Programs

Due to the length of the LLVM code generated, the code for the target programs are at the end of this section.

6.1.1 Rippl in One Slide

Source program: `onehotrippl.rpl`

```
map = fun f -> fun list -> if len list == 0
        then []
        else f~(head list) cons (map~f~(tail list))

main :: [(int,int)]
main = let l = map~(fun x -> x+3)~[2,5,66] in
    [ (y,y+1) | y over l, y > 10 ]
```

Output: `[(69, 70)]`

6.1.2 Mersenne Primes

Source program: `mersenne.rpl`

```
divides = fun a -> fun b -> if (a % b == 0) then true else false

is_prime = fun n -> if n < 2 then false else
                let lst = [2...n] in
                let d = [ z | z over lst, (divides~n~z) ] in
                if head d == n then true else false

mersenne = fun i -> [2^x-1 | x over [1...i], is_prime~(2^x-1)]

main :: [int]
main = mersenne~10
```

Output: `[3, 7, 31, 127]`

6.1.3 Collatz Conjecture

Source program: `collatz.rpl`

```
collatz = fun x ->
    let rec_collatz = fun n -> fun l ->
        if n == 1
            then 1 cons l
            else if (n % 2 == 0)
                then rec_collatz~(n / 2)~(n cons l)
                else rec_collatz~(3*n + 1)~(n cons l)
    in rec_collatz~x~[]
main :: [int]
main = collatz~99
```

Output: `[1, 2, 4, 8, 16, 5, 10, 20, 40, 13, 26, 52, 17, 34, 11, 22, 7, 14, 28, 56, 112, 224, 448, 149, 298, 99]`

6.2 Test Scripts

We have one test script that runs all of the `.rpl` files in the current directory and checks their exit status code.

Test script: `run_tests.sh`

```
#!/bin/zsh
#assert.sh

for file in *.rpl
do
    rippl $file &> /dev/null
    if [ $? -ne 0 ]
    then echo -e "\e[39m$file \e[91mfailed"
    else  echo -e "\e[39m$file \e[92mpassed"
    fi

    ./${file: : -4} &> /dev/null
    if [ $? -ne 0 ]
    then echo -e "\e[39m$file \e[91mfailed"
    else echo -e "\e[39m$file \e[92mpassed"
    fi

    rm -rf ${file: : -4} &> /dev/null
        rm -rf *.byte *.byte.s &> /dev/null
done
```

The automation of testing is done by iterating over all source files in a given directory, building them, and running them. For each file tested, there is a success or failure message, one for compilation and execution. Here is a sample run of the test script:

```
./run_tests.sh
arrowlist.rpl passed
arrowlist.rpl passed
collatz.rpl passed
collatz.rpl passed
fibonacci.rpl passed
fibonacci.rpl passed
fun_tuple.rpl passed
fun_tuple.rpl passed
hello.rpl passed
hello.rpl passed
hollis.rpl passed
hollis.rpl passed
ite.rpl passed
ite.rpl passed
just.rpl passed
just.rpl passed
lamrange.rpl passed
lamrange.rpl passed
lazy.rpl passed
lazy.rpl passed
listcomp.rpl passed
listcomp.rpl passed
mersenne.rpl passed
mersenne.rpl passed
mymap.rpl passed
mymap.rpl passed
nSquares.rpl passed
nSquares.rpl passed
onehotrippl.rpl passed
onehotrippl.rpl passed
partapp.rpl passed
partapp.rpl passed
plthw1.rpl passed
plthw1.rpl passed
range.rpl passed
range.rpl passed
ronghui-nl.rpl passed
ronghui-nl.rpl passed
skip.rpl passed
skip.rpl passed
tuple_square.rpl passed
tuple_square.rpl passed
tuptest.rpl passed
tuptest.rpl passed
```

6.2.1 Who Did What?

Amanda and Hollis wrote tests for type inference, Hans and Da wrote tests for lambda lifting, and everyone wrote a few tests for code generation and general testing when the compiler was functional.

6.2.2 LLVM IR for test programs

onehotrippl.byte

```
; ModuleID = 'Rippl'
source_filename = "Rippl"

%Thunk = type { i8* (%Thunk*)*, i32, i32, %Thunk**, i8*, i32 }
%Node = type { %Thunk*, %Node* }
%List = type { %Node*, i32, i32, %Node*, i32, i32, i32 }

@fmt = private unnamed_addr constant [3 x i8] c"%c\00"
@fmt_int = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt_float = private unnamed_addr constant [3 x i8] c"%f\00"
@add_init_thunk = global %Thunk zeroinitializer, align 32
@sub_init_thunk = global %Thunk zeroinitializer, align 32
@mult_init_thunk = global %Thunk zeroinitializer, align 32
@divi_init_thunk = global %Thunk zeroinitializer, align 32
@mod_init_thunk = global %Thunk zeroinitializer, align 32
@powe_init_thunk = global %Thunk zeroinitializer, align 32
@eq_init_thunk = global %Thunk zeroinitializer, align 32
@neq_init_thunk = global %Thunk zeroinitializer, align 32
@geq_init_thunk = global %Thunk zeroinitializer, align 32
@leq_init_thunk = global %Thunk zeroinitializer, align 32
@less_init_thunk = global %Thunk zeroinitializer, align 32
@greater_init_thunk = global %Thunk zeroinitializer, align 32
@neg_init_thunk = global %Thunk zeroinitializer, align 32
@addf_init_thunk = global %Thunk zeroinitializer, align 32
@subf_init_thunk = global %Thunk zeroinitializer, align 32
@multf_init_thunk = global %Thunk zeroinitializer, align 32
@divf_init_thunk = global %Thunk zeroinitializer, align 32
@powef_init_thunk = global %Thunk zeroinitializer, align 32
@eqf_init_thunk = global %Thunk zeroinitializer, align 32
@neqf_init_thunk = global %Thunk zeroinitializer, align 32
@geqf_init_thunk = global %Thunk zeroinitializer, align 32
@leqf_init_thunk = global %Thunk zeroinitializer, align 32
@lessf_init_thunk = global %Thunk zeroinitializer, align 32
@greaterf_init_thunk = global %Thunk zeroinitializer, align 32
@negf_init_thunk = global %Thunk zeroinitializer, align 32
@andb_init_thunk = global %Thunk zeroinitializer, align 32
@orb_init_thunk = global %Thunk zeroinitializer, align 32
@notb_init_thunk = global %Thunk zeroinitializer, align 32
@cons_init_thunk = global %Thunk zeroinitializer, align 32
@cat_init_thunk = global %Thunk zeroinitializer, align 32
@length_init_thunk = global %Thunk zeroinitializer, align 32
@head_init_thunk = global %Thunk zeroinitializer, align 32
@tail_init_thunk = global %Thunk zeroinitializer, align 32
@is_none_init_thunk = global %Thunk zeroinitializer, align 32
@from_just_init_thunk = global %Thunk zeroinitializer, align 32
@first_init_thunk = global %Thunk zeroinitializer, align 32
@second_init_thunk = global %Thunk zeroinitializer, align 32
@ite_init_thunk = global %Thunk zeroinitializer, align 32
```

```
@int_to_float_init_thunk = global %Thunk zeroinitializer, align 32
@Thunk = external global %Thunk
@"$$map_init_thunk" = global %Thunk zeroinitializer
@"$$$anon2_init_thunk" = global %Thunk zeroinitializer
@"$$$anon3_init_thunk" = global %Thunk zeroinitializer
@"$$$anon4_init_thunk" = global %Thunk zeroinitializer

define i32 @main() {
entry:
  call void @initNativeThunks()
  %initThunk = call %Thunk* @init_thunk(%Thunk* @"$$map_init_thunk",
     i8* (%Thunk*)* @"$eval_map", i32 2, i32 0)
  %initThunk1 = call %Thunk* @init_thunk(%Thunk* @"$$$anon2_init_thunk",
    i8* (%Thunk*)* @"$eval_$anon2", i32 1, i32 0)
  %initThunk2 = call %Thunk* @init_thunk(%Thunk* @"$$$anon3_init_thunk",
    i8* (%Thunk*)* @"$eval_$anon3", i32 1, i32 0)
  %initThunk3 = call %Thunk* @init_thunk(%Thunk* @"$$$anon4_init_thunk",
    i8* (%Thunk*)* @"$eval_$anon4", i32 1, i32 0)
  %ty_heap = alloca i32, i32 7
  %pos = getelementptr i32, i32* %ty_heap, i32 0
  store i32 4, i32* %pos
  %pos4 = getelementptr i32, i32* %ty_heap, i32 1
  store i32 5, i32* %pos4
  %pos5 = getelementptr i32, i32* %ty_heap, i32 3
  store i32 0, i32* %pos5
  %pos6 = getelementptr i32, i32* %ty_heap, i32 4
  store i32 0, i32* %pos6
  %apply = call %Thunk* @apply(%Thunk* @"$$map_init_thunk", %Thunk*
    @"$$$anon2_init_thunk")
  %empty = call %Thunk* @makeEmptyList(i32 0)
  %makeInt = call %Thunk* @makeInt(i32 2)
  %makeNode = call %Node* @makeNode(%Thunk* %makeInt)
  %appendNodeThunk = call %Thunk* @appendNodeThunk(%Thunk* %empty,
    %Node* %makeNode)
  %makeInt7 = call %Thunk* @makeInt(i32 5)
  %makeNode8 = call %Node* @makeNode(%Thunk* %makeInt7)
  %appendNodeThunk9 = call %Thunk* @appendNodeThunk(%Thunk*
    %appendNodeThunk, %Node* %makeNode8)
  %makeInt10 = call %Thunk* @makeInt(i32 66)
  %makeNode11 = call %Node* @makeNode(%Thunk* %makeInt10)
  %appendNodeThunk12 = call %Thunk* @appendNodeThunk(%Thunk*
    %appendNodeThunk9, %Node* %makeNode11)
  %apply13 = call %Thunk* @apply(%Thunk* %apply, %Thunk*
    %appendNodeThunk12)
  %hans = alloca %Thunk*
  store %Thunk* %apply13, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %map_thunk = call %Thunk* @mapl(%Thunk* %loaded_hans, %Thunk*
    @"$$$anon3_init_thunk", i32 5)
  %filter_thunk = call %Thunk* @filterl(%Thunk* %map_thunk, %Thunk*
    @"$$$anon4_init_thunk", i32 5)
```

```
  %invoke = call i8* @invoke(%Thunk* %filter_thunk)
  call void @printAnyThunk(%Thunk* %filter_thunk, i32* %ty_heap, i32 0)
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds
    ([3 x i8], [3 x i8]* @fmt, i32 0, i32 0), i8 10)
  ret i32 0
}

declare %Thunk* @makeInt(i32)

declare %Thunk* @makeBool(i8)

declare %Thunk* @makeChar(i8)

declare %Thunk* @makeFloat(float)

declare %Thunk* @makeTuple(%Thunk*, %Thunk*, i32, i32)

declare %Thunk* @makeMaybe(%Thunk*, i32)

declare %Node* @makeNode(%Thunk*)

declare %Thunk* @makeEmptyList(i32)

declare %Thunk* @makeInfinite(i32)

declare %Thunk* @makeRangeList(%Thunk*, %Thunk*)

declare %List* @appendNode(%List*, %Node*)

declare %Thunk* @appendNodeThunk(%Thunk*, %Node*)

declare i32 @printf(i8*, ...)

declare void @printRangeList(%Thunk*)

declare void @printPrim(i8*, i32)

declare void @printAnyThunk(%Thunk*, i32*, i32)

declare void @printBool(i8)

declare void @initNativeThunks()

declare %Thunk* @mapl(%Thunk*, %Thunk*, i32)

declare %Thunk* @filterl(%Thunk*, %Thunk*, i32)

declare %Thunk* @map_listl(%Thunk*, %Thunk*, i32)

declare i32* @add(%Thunk*, %Thunk*)
```

```
declare i8* @add_eval(%Thunk*)

declare i32* @sub(%Thunk*, %Thunk*)

declare i8* @sub_eval(%Thunk*)

declare i32* @mult(%Thunk*, %Thunk*)

declare i8* @mult_eval(%Thunk*)

declare i32* @neq(%Thunk*, %Thunk*)

declare i8* @neq_eval(%Thunk*)

declare i8* @addf_eval(%Thunk*)

declare i32* @addf(%Thunk*, %Thunk*)

declare %Thunk* @init_thunk(%Thunk*, i8* (%Thunk*)*, i32, i32)

declare %Thunk* @init_thunk_literal(i8*)

declare %Thunk* @apply(%Thunk*, %Thunk*)

declare i8* @invoke(%Thunk*)

define i8* @"$eval_map"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %thunk2 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %tload2 = load %Thunk*, %Thunk** %pthunk
  %args3 = getelementptr %Thunk, %Thunk* %tload2, i32 0, i32 3
  %loadargs4 = load %Thunk**, %Thunk*** %args3
  %args5 = getelementptr %Thunk*, %Thunk** %loadargs4, i32 1
  %loadarg6 = load %Thunk*, %Thunk** %args5
  store %Thunk* %loadarg6, %Thunk** %thunk2
  %load = load %Thunk*, %Thunk** %thunk2
  %load7 = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @map(%Thunk* %load7, %Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
```

```
}

define i8* @"$eval_$anon2"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @"$anon2"(%Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_$anon3"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @"$anon3"(%Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_$anon4"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
```

```
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @"$anon4"(%Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @map(%Thunk*, %Thunk*) {
entry:
  %f = alloca %Thunk*
  store %Thunk* %0, %Thunk** %f
  %amanda = load %Thunk*, %Thunk** %f
  %list = alloca %Thunk*
  store %Thunk* %1, %Thunk** %list
  %amanda1 = load %Thunk*, %Thunk** %list
  %hans = alloca %Thunk*
  store %Thunk* %amanda1, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply = call %Thunk* @apply(%Thunk* @length_init_thunk, %Thunk* %loaded_hans)
  %apply2 = call %Thunk* @apply(%Thunk* @eq_init_thunk, %Thunk* %apply)
  %makeInt = call %Thunk* @makeInt(i32 0)
  %apply3 = call %Thunk* @apply(%Thunk* %apply2, %Thunk* %makeInt)
  %empty = call %Thunk* @makeEmptyList(i32 -1)
  %hans4 = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans4
  %loaded_hans5 = load %Thunk*, %Thunk** %hans4
  %hans6 = alloca %Thunk*
  store %Thunk* %amanda1, %Thunk** %hans6
  %loaded_hans7 = load %Thunk*, %Thunk** %hans6
  %apply8 = call %Thunk* @apply(%Thunk* @head_init_thunk, %Thunk*
    %loaded_hans7)
  %apply9 = call %Thunk* @apply(%Thunk* %loaded_hans5, %Thunk* %apply8)
  %apply10 = call %Thunk* @apply(%Thunk* @cons_init_thunk, %Thunk*
  %apply9)
  %hans11 = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans11
  %loaded_hans12 = load %Thunk*, %Thunk** %hans11
  %apply13 = call %Thunk* @apply(%Thunk* @"$$map_init_thunk", %Thunk*
  %loaded_hans12)
  %hans14 = alloca %Thunk*
  store %Thunk* %amanda1, %Thunk** %hans14
  %loaded_hans15 = load %Thunk*, %Thunk** %hans14
  %apply16 = call %Thunk* @apply(%Thunk* @tail_init_thunk, %Thunk*
  %loaded_hans15)
  %apply17 = call %Thunk* @apply(%Thunk* %apply13, %Thunk* %apply16)
  %apply18 = call %Thunk* @apply(%Thunk* %apply10, %Thunk* %apply17)
  %apply19 = call %Thunk* @apply(%Thunk* @ite_init_thunk, %Thunk*
  %apply3)
  %apply20 = call %Thunk* @apply(%Thunk* %apply19, %Thunk* %empty)
  %apply21 = call %Thunk* @apply(%Thunk* %apply20, %Thunk* %apply18)
```

```
  %da = call i8* @invoke(%Thunk* %apply21)
  ret i8* %da
}

define i8* @"$anon2"(%Thunk*) {
entry:
  %x = alloca %Thunk*
  store %Thunk* %0, %Thunk** %x
  %amanda = load %Thunk*, %Thunk** %x
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply = call %Thunk* @apply(%Thunk* @add_init_thunk, %Thunk*
  %loaded_hans)
  %makeInt = call %Thunk* @makeInt(i32 3)
  %apply1 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %makeInt)
  %da = call i8* @invoke(%Thunk* %apply1)
  ret i8* %da
}

define i8* @"$anon3"(%Thunk*) {
entry:
  %"$y1" = alloca %Thunk*
  store %Thunk* %0, %Thunk** %"$y1"
  %amanda = load %Thunk*, %Thunk** %"$y1"
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %hans1 = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans1
  %loaded_hans2 = load %Thunk*, %Thunk** %hans1
  %apply = call %Thunk* @apply(%Thunk* @add_init_thunk, %Thunk*
  %loaded_hans2)
  %makeInt = call %Thunk* @makeInt(i32 1)
  %apply3 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %makeInt)
  %tup = call %Thunk* @makeTuple(%Thunk* %loaded_hans, %Thunk* %apply3,
  i32 0, i32 0)
  %da = call i8* @invoke(%Thunk* %tup)
  ret i8* %da
}

define i8* @"$anon4"(%Thunk*) {
entry:
  %"$y0" = alloca %Thunk*
  store %Thunk* %0, %Thunk** %"$y0"
  %amanda = load %Thunk*, %Thunk** %"$y0"
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply = call %Thunk* @apply(%Thunk* @greater_init_thunk, %Thunk*
  %loaded_hans)
```

```
  %makeInt = call %Thunk* @makeInt(i32 10)
  %apply1 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %makeInt)
  %da = call i8* @invoke(%Thunk* %apply1)
  ret i8* %da
}
```

mersenne.byte

```
; ModuleID = 'Rippl'
source_filename = "Rippl"

%Thunk = type { i8* (%Thunk*)*, i32, i32, %Thunk**, i8*, i32 }
%Node = type { %Thunk*, %Node* }
%List = type { %Node*, i32, i32, %Node*, i32, i32, i32 }

@fmt = private unnamed_addr constant [3 x i8] c"%c\00"
@fmt_int = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt_float = private unnamed_addr constant [3 x i8] c"%f\00"
@add_init_thunk = global %Thunk zeroinitializer, align 32
@sub_init_thunk = global %Thunk zeroinitializer, align 32
@mult_init_thunk = global %Thunk zeroinitializer, align 32
@divi_init_thunk = global %Thunk zeroinitializer, align 32
@mod_init_thunk = global %Thunk zeroinitializer, align 32
@powe_init_thunk = global %Thunk zeroinitializer, align 32
@eq_init_thunk = global %Thunk zeroinitializer, align 32
@neq_init_thunk = global %Thunk zeroinitializer, align 32
@geq_init_thunk = global %Thunk zeroinitializer, align 32
@leq_init_thunk = global %Thunk zeroinitializer, align 32
@less_init_thunk = global %Thunk zeroinitializer, align 32
@greater_init_thunk = global %Thunk zeroinitializer, align 32
@neg_init_thunk = global %Thunk zeroinitializer, align 32
@addf_init_thunk = global %Thunk zeroinitializer, align 32
@subf_init_thunk = global %Thunk zeroinitializer, align 32
@multf_init_thunk = global %Thunk zeroinitializer, align 32
@divf_init_thunk = global %Thunk zeroinitializer, align 32
@powef_init_thunk = global %Thunk zeroinitializer, align 32
@eqf_init_thunk = global %Thunk zeroinitializer, align 32
@neqf_init_thunk = global %Thunk zeroinitializer, align 32
@geqf_init_thunk = global %Thunk zeroinitializer, align 32
@leqf_init_thunk = global %Thunk zeroinitializer, align 32
@lessf_init_thunk = global %Thunk zeroinitializer, align 32
@greaterf_init_thunk = global %Thunk zeroinitializer, align 32
@negf_init_thunk = global %Thunk zeroinitializer, align 32
@andb_init_thunk = global %Thunk zeroinitializer, align 32
@orb_init_thunk = global %Thunk zeroinitializer, align 32
@notb_init_thunk = global %Thunk zeroinitializer, align 32
@cons_init_thunk = global %Thunk zeroinitializer, align 32
@cat_init_thunk = global %Thunk zeroinitializer, align 32
@length_init_thunk = global %Thunk zeroinitializer, align 32
@head_init_thunk = global %Thunk zeroinitializer, align 32
@tail_init_thunk = global %Thunk zeroinitializer, align 32
@is_none_init_thunk = global %Thunk zeroinitializer, align 32
@from_just_init_thunk = global %Thunk zeroinitializer, align 32
```

```
@first_init_thunk = global %Thunk zeroinitializer, align 32
@second_init_thunk = global %Thunk zeroinitializer, align 32
@ite_init_thunk = global %Thunk zeroinitializer, align 32
@int_to_float_init_thunk = global %Thunk zeroinitializer, align 32
@Thunk = external global %Thunk
@"$$divides_init_thunk" = global %Thunk zeroinitializer
@"$$$anon2_init_thunk" = global %Thunk zeroinitializer
@"$$$anon3_init_thunk" = global %Thunk zeroinitializer
@"$$is_prime_init_thunk" = global %Thunk zeroinitializer
@"$$$anon7_init_thunk" = global %Thunk zeroinitializer
@"$$$anon8_init_thunk" = global %Thunk zeroinitializer
@"$$mersenne_init_thunk" = global %Thunk zeroinitializer

define i32 @main() {
entry:
  call void @initNativeThunks()
  %initThunk = call %Thunk* @init_thunk(%Thunk* @"$$divides_init_thunk",
    i8* (%Thunk*)* @"$eval_divides", i32 2, i32 0)
  %initThunk1 = call %Thunk* @init_thunk(%Thunk* @"$$$anon2_init_thunk",
    i8* (%Thunk*)* @"$eval_$anon2", i32 1, i32 0)
  %initThunk2 = call %Thunk* @init_thunk(%Thunk* @"$$$anon3_init_thunk",
    i8* (%Thunk*)* @"$eval_$anon3", i32 2, i32 0)
  %initThunk3 = call %Thunk* @init_thunk(%Thunk*
    @"$$is_prime_init_thunk", i8* (%Thunk*)* @"$eval_is_prime",
    i32 1, i32 0)
  %initThunk4 = call %Thunk* @init_thunk(%Thunk* @"$$$anon7_init_thunk",
    i8* (%Thunk*)* @"$eval_$anon7", i32 1, i32 0)
  %initThunk5 = call %Thunk* @init_thunk(%Thunk* @"$$$anon8_init_thunk",
    i8* (%Thunk*)* @"$eval_$anon8", i32 1, i32 0)
  %initThunk6 = call %Thunk* @init_thunk(%Thunk*
    @"$$mersenne_init_thunk", i8* (%Thunk*)* @"$eval_mersenne", i32 1,
    i32 0)
  %ty_heap = alloca i32, i32 3
  %pos = getelementptr i32, i32* %ty_heap, i32 0
  store i32 4, i32* %pos
  %pos7 = getelementptr i32, i32* %ty_heap, i32 1
  store i32 0, i32* %pos7
  %makeInt = call %Thunk* @makeInt(i32 10)
  %apply = call %Thunk* @apply(%Thunk* @"$$mersenne_init_thunk",
    %Thunk* %makeInt)
  %invoke = call i8* @invoke(%Thunk* %apply)
  call void @printAnyThunk(%Thunk* %apply, i32* %ty_heap, i32 0)
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds
    ([3 x i8], [3 x i8]* @fmt, i32 0, i32 0), i8 10)
  ret i32 0
}

declare %Thunk* @makeInt(i32)

declare %Thunk* @makeBool(i8)
```

```
declare %Thunk* @makeChar(i8)

declare %Thunk* @makeFloat(float)

declare %Thunk* @makeTuple(%Thunk*, %Thunk*, i32, i32)

declare %Thunk* @makeMaybe(%Thunk*, i32)

declare %Node* @makeNode(%Thunk*)

declare %Thunk* @makeEmptyList(i32)

declare %Thunk* @makeInfinite(i32)

declare %Thunk* @makeRangeList(%Thunk*, %Thunk*)

declare %List* @appendNode(%List*, %Node*)

declare %Thunk* @appendNodeThunk(%Thunk*, %Node*)

declare i32 @printf(i8*, ...)

declare void @printRangeList(%Thunk*)

declare void @printPrim(i8*, i32)

declare void @printAnyThunk(%Thunk*, i32*, i32)

declare void @printBool(i8)

declare void @initNativeThunks()

declare %Thunk* @mapl(%Thunk*, %Thunk*, i32)

declare %Thunk* @filterl(%Thunk*, %Thunk*, i32)

declare %Thunk* @map_listl(%Thunk*, %Thunk*, i32)

declare i32* @add(%Thunk*, %Thunk*)

declare i8* @add_eval(%Thunk*)

declare i32* @sub(%Thunk*, %Thunk*)

declare i8* @sub_eval(%Thunk*)

declare i32* @mult(%Thunk*, %Thunk*)

declare i8* @mult_eval(%Thunk*)

declare i32* @neq(%Thunk*, %Thunk*)
```

```
declare i8* @neq_eval(%Thunk*)

declare i8* @addf_eval(%Thunk*)

declare i32* @addf(%Thunk*, %Thunk*)

declare %Thunk* @init_thunk(%Thunk*, i8* (%Thunk*)*, i32, i32)

declare %Thunk* @init_thunk_literal(i8*)

declare %Thunk* @apply(%Thunk*, %Thunk*)

declare i8* @invoke(%Thunk*)

define i8* @"$eval_divides"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %thunk2 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %tload2 = load %Thunk*, %Thunk** %pthunk
  %args3 = getelementptr %Thunk, %Thunk* %tload2, i32 0, i32 3
  %loadargs4 = load %Thunk**, %Thunk*** %args3
  %args5 = getelementptr %Thunk*, %Thunk** %loadargs4, i32 1
  %loadarg6 = load %Thunk*, %Thunk** %args5
  store %Thunk* %loadarg6, %Thunk** %thunk2
  %load = load %Thunk*, %Thunk** %thunk2
  %load7 = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @divides(%Thunk* %load7, %Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_$anon2"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
```

```
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @"$anon2"(%Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_$anon3"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %thunk2 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %tload2 = load %Thunk*, %Thunk** %pthunk
  %args3 = getelementptr %Thunk, %Thunk* %tload2, i32 0, i32 3
  %loadargs4 = load %Thunk**, %Thunk*** %args3
  %args5 = getelementptr %Thunk*, %Thunk** %loadargs4, i32 1
  %loadarg6 = load %Thunk*, %Thunk** %args5
  store %Thunk* %loadarg6, %Thunk** %thunk2
  %load = load %Thunk*, %Thunk** %thunk2
  %load7 = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @"$anon3"(%Thunk* %load7, %Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_is_prime"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @is_prime(%Thunk* %load)
```

```
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_$anon7"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @"$anon7"(%Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_$anon8"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @"$anon8"(%Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_mersenne"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
```

```
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @mersenne(%Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @divides(%Thunk*, %Thunk*) {
entry:
  %a = alloca %Thunk*
  store %Thunk* %0, %Thunk** %a
  %amanda = load %Thunk*, %Thunk** %a
  %b = alloca %Thunk*
  store %Thunk* %1, %Thunk** %b
  %amanda1 = load %Thunk*, %Thunk** %b
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply = call %Thunk* @apply(%Thunk* @mod_init_thunk, %Thunk* %loaded_hans)
  %hans2 = alloca %Thunk*
  store %Thunk* %amanda1, %Thunk** %hans2
  %loaded_hans3 = load %Thunk*, %Thunk** %hans2
  %apply4 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %loaded_hans3)
  %apply5 = call %Thunk* @apply(%Thunk* @eq_init_thunk, %Thunk* %apply4)
  %makeInt = call %Thunk* @makeInt(i32 0)
  %apply6 = call %Thunk* @apply(%Thunk* %apply5, %Thunk* %makeInt)
  %makeBool = call %Thunk* @makeBool(i8 1)
  %makeBool7 = call %Thunk* @makeBool(i8 0)
  %apply8 = call %Thunk* @apply(%Thunk* @ite_init_thunk, %Thunk*
    %apply6)
  %apply9 = call %Thunk* @apply(%Thunk* %apply8, %Thunk* %makeBool)
  %apply10 = call %Thunk* @apply(%Thunk* %apply9, %Thunk* %makeBool7)
  %da = call i8* @invoke(%Thunk* %apply10)
  ret i8* %da
}

define i8* @"$anon2"(%Thunk*) {
entry:
  %"$z1" = alloca %Thunk*
  store %Thunk* %0, %Thunk** %"$z1"
  %amanda = load %Thunk*, %Thunk** %"$z1"
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %da = call i8* @invoke(%Thunk* %loaded_hans)
  ret i8* %da
}
```

```
define i8* @"$anon3"(%Thunk*, %Thunk*) {
entry:
  %"$n4" = alloca %Thunk*
  store %Thunk* %0, %Thunk** %"$n4"
  %amanda = load %Thunk*, %Thunk** %"$n4"
  %"$z0" = alloca %Thunk*
  store %Thunk* %1, %Thunk** %"$z0"
  %amanda1 = load %Thunk*, %Thunk** %"$z0"
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply = call %Thunk* @apply(%Thunk* @"$$divides_init_thunk",
    %Thunk* %loaded_hans)
  %hans2 = alloca %Thunk*
  store %Thunk* %amanda1, %Thunk** %hans2
  %loaded_hans3 = load %Thunk*, %Thunk** %hans2
  %apply4 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %loaded_hans3)
  %da = call i8* @invoke(%Thunk* %apply4)
  ret i8* %da
}

define i8* @is_prime(%Thunk*) {
entry:
  %n = alloca %Thunk*
  store %Thunk* %0, %Thunk** %n
  %amanda = load %Thunk*, %Thunk** %n
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply = call %Thunk* @apply(%Thunk* @less_init_thunk,
    %Thunk* %loaded_hans)
  %makeInt = call %Thunk* @makeInt(i32 2)
  %apply1 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %makeInt)
  %makeBool = call %Thunk* @makeBool(i8 0)
  %makeInt2 = call %Thunk* @makeInt(i32 2)
  %hans3 = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans3
  %loaded_hans4 = load %Thunk*, %Thunk** %hans3
  %range = call %Thunk* @makeRangeList(%Thunk* %makeInt2, %Thunk* %loaded_hans4)
  %hans5 = alloca %Thunk*
  store %Thunk* %range, %Thunk** %hans5
  %loaded_hans6 = load %Thunk*, %Thunk** %hans5
  %hans7 = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans7
  %loaded_hans8 = load %Thunk*, %Thunk** %hans7
  %apply9 = call %Thunk* @apply(%Thunk* @"$$$anon3_init_thunk",
    %Thunk* %loaded_hans8)
  %map_thunk = call %Thunk* @mapl(%Thunk* %loaded_hans6,
    %Thunk* @"$$$anon2_init_thunk", i32 0)
  %filter_thunk = call %Thunk* @filterl(%Thunk* %map_thunk,
```

```
    %Thunk* %apply9, i32 0)
  %hans10 = alloca %Thunk*
  store %Thunk* %filter_thunk, %Thunk** %hans10
  %loaded_hans11 = load %Thunk*, %Thunk** %hans10
  %apply12 = call %Thunk* @apply(%Thunk* @head_init_thunk,
    %Thunk* %loaded_hans11)
  %apply13 = call %Thunk* @apply(%Thunk* @eq_init_thunk, %Thunk* %apply12)
  %hans14 = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans14
  %loaded_hans15 = load %Thunk*, %Thunk** %hans14
  %apply16 = call %Thunk* @apply(%Thunk* %apply13,
    %Thunk* %loaded_hans15)
  %makeBool17 = call %Thunk* @makeBool(i8 1)
  %makeBool18 = call %Thunk* @makeBool(i8 0)
  %apply19 = call %Thunk* @apply(%Thunk* @ite_init_thunk,
    %Thunk* %apply16)
  %apply20 = call %Thunk* @apply(%Thunk* %apply19, %Thunk* %makeBool17)
  %apply21 = call %Thunk* @apply(%Thunk* %apply20, %Thunk* %makeBool18)
  %apply22 = call %Thunk* @apply(%Thunk* @ite_init_thunk,
    %Thunk* %apply1)
  %apply23 = call %Thunk* @apply(%Thunk* %apply22, %Thunk* %makeBool)
  %apply24 = call %Thunk* @apply(%Thunk* %apply23, %Thunk* %apply21)
  %da = call i8* @invoke(%Thunk* %apply24)
  ret i8* %da
}

define i8* @"$anon7"(%Thunk*) {
entry:
  %"$x6" = alloca %Thunk*
  store %Thunk* %0, %Thunk** %"$x6"
  %amanda = load %Thunk*, %Thunk** %"$x6"
  %makeInt = call %Thunk* @makeInt(i32 2)
  %apply = call %Thunk* @apply(%Thunk* @powe_init_thunk,
    %Thunk* %makeInt)
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply1 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %loaded_hans)
  %apply2 = call %Thunk* @apply(%Thunk* @sub_init_thunk,
    %Thunk* %apply1)
  %makeInt3 = call %Thunk* @makeInt(i32 1)
  %apply4 = call %Thunk* @apply(%Thunk* %apply2, %Thunk* %makeInt3)
  %da = call i8* @invoke(%Thunk* %apply4)
  ret i8* %da
}

define i8* @"$anon8"(%Thunk*) {
entry:
  %"$x5" = alloca %Thunk*
  store %Thunk* %0, %Thunk** %"$x5"
  %amanda = load %Thunk*, %Thunk** %"$x5"
```

```
  %makeInt = call %Thunk* @makeInt(i32 2)
  %apply = call %Thunk* @apply(%Thunk* @powe_init_thunk,
    %Thunk* %makeInt)
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply1 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %loaded_hans)
  %apply2 = call %Thunk* @apply(%Thunk* @sub_init_thunk,
    %Thunk* %apply1)
  %makeInt3 = call %Thunk* @makeInt(i32 1)
  %apply4 = call %Thunk* @apply(%Thunk* %apply2, %Thunk* %makeInt3)
  %apply5 = call %Thunk* @apply(%Thunk* @"$$is_prime_init_thunk",
    %Thunk* %apply4)
  %da = call i8* @invoke(%Thunk* %apply5)
  ret i8* %da
}

define i8* @mersenne(%Thunk*) {
entry:
  %i = alloca %Thunk*
  store %Thunk* %0, %Thunk** %i
  %amanda = load %Thunk*, %Thunk** %i
  %makeInt = call %Thunk* @makeInt(i32 1)
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %range = call %Thunk* @makeRangeList(%Thunk* %makeInt, %Thunk*
    %loaded_hans)
  %map_thunk = call %Thunk* @mapl(%Thunk* %range, %Thunk*
    @"$$$anon7_init_thunk", i32 0)
  %filter_thunk = call %Thunk* @filterl(%Thunk* %map_thunk, %Thunk*
    @"$$$anon8_init_thunk", i32 0)
  %da = call i8* @invoke(%Thunk* %filter_thunk)
  ret i8* %da
}
```

collatz.byte

```
; ModuleID = 'Rippl'
source_filename = "Rippl"

%Thunk = type { i8* (%Thunk*)*, i32, i32, %Thunk**, i8*, i32 }
%Node = type { %Thunk*, %Node* }
%List = type { %Node*, i32, i32, %Node*, i32, i32, i32 }

@fmt = private unnamed_addr constant [3 x i8] c"%c\00"
@fmt_int = private unnamed_addr constant [3 x i8] c"%d\00"
@fmt_float = private unnamed_addr constant [3 x i8] c"%f\00"
@add_init_thunk = global %Thunk zeroinitializer, align 32
@sub_init_thunk = global %Thunk zeroinitializer, align 32
@mult_init_thunk = global %Thunk zeroinitializer, align 32
@divi_init_thunk = global %Thunk zeroinitializer, align 32
@mod_init_thunk = global %Thunk zeroinitializer, align 32
```

```
@powe_init_thunk = global %Thunk zeroinitializer, align 32
@eq_init_thunk = global %Thunk zeroinitializer, align 32
@neq_init_thunk = global %Thunk zeroinitializer, align 32
@geq_init_thunk = global %Thunk zeroinitializer, align 32
@leq_init_thunk = global %Thunk zeroinitializer, align 32
@less_init_thunk = global %Thunk zeroinitializer, align 32
@greater_init_thunk = global %Thunk zeroinitializer, align 32
@neg_init_thunk = global %Thunk zeroinitializer, align 32
@addf_init_thunk = global %Thunk zeroinitializer, align 32
@subf_init_thunk = global %Thunk zeroinitializer, align 32
@multf_init_thunk = global %Thunk zeroinitializer, align 32
@divf_init_thunk = global %Thunk zeroinitializer, align 32
@powef_init_thunk = global %Thunk zeroinitializer, align 32
@eqf_init_thunk = global %Thunk zeroinitializer, align 32
@neqf_init_thunk = global %Thunk zeroinitializer, align 32
@geqf_init_thunk = global %Thunk zeroinitializer, align 32
@leqf_init_thunk = global %Thunk zeroinitializer, align 32
@lessf_init_thunk = global %Thunk zeroinitializer, align 32
@greaterf_init_thunk = global %Thunk zeroinitializer, align 32
@negf_init_thunk = global %Thunk zeroinitializer, align 32
@andb_init_thunk = global %Thunk zeroinitializer, align 32
@orb_init_thunk = global %Thunk zeroinitializer, align 32
@notb_init_thunk = global %Thunk zeroinitializer, align 32
@cons_init_thunk = global %Thunk zeroinitializer, align 32
@cat_init_thunk = global %Thunk zeroinitializer, align 32
@length_init_thunk = global %Thunk zeroinitializer, align 32
@head_init_thunk = global %Thunk zeroinitializer, align 32
@tail_init_thunk = global %Thunk zeroinitializer, align 32
@is_none_init_thunk = global %Thunk zeroinitializer, align 32
@from_just_init_thunk = global %Thunk zeroinitializer, align 32
@first_init_thunk = global %Thunk zeroinitializer, align 32
@second_init_thunk = global %Thunk zeroinitializer, align 32
@ite_init_thunk = global %Thunk zeroinitializer, align 32
@int_to_float_init_thunk = global %Thunk zeroinitializer, align 32
@Thunk = external global %Thunk
@"$$rec_collatz_init_thunk" = global %Thunk zeroinitializer
@"$$collatz_init_thunk" = global %Thunk zeroinitializer

define i32 @main() {
entry:
  call void @initNativeThunks()
  %initThunk = call %Thunk* @init_thunk(%Thunk*
    @"$$rec_collatz_init_thunk", i8* (%Thunk*)* @"$eval_rec_collatz",
    i32 2, i32 0)
  %initThunk1 = call %Thunk* @init_thunk(%Thunk*
    @"$$collatz_init_thunk", i8* (%Thunk*)* @"$eval_collatz", i32 1,
    i32 0)
  %ty_heap = alloca i32, i32 3
  %pos = getelementptr i32, i32* %ty_heap, i32 0
  store i32 4, i32* %pos
  %pos2 = getelementptr i32, i32* %ty_heap, i32 1
```

```
  store i32 0, i32* %pos2
  %makeInt = call %Thunk* @makeInt(i32 99)
  %apply = call %Thunk* @apply(%Thunk* @"$$collatz_init_thunk",
    %Thunk* %makeInt)
  %invoke = call i8* @invoke(%Thunk* %apply)
  call void @printAnyThunk(%Thunk* %apply, i32* %ty_heap, i32 0)
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds
    ([3 x i8], [3 x i8]* @fmt, i32 0, i32 0), i8 10)
  ret i32 0
}

declare %Thunk* @makeInt(i32)

declare %Thunk* @makeBool(i8)

declare %Thunk* @makeChar(i8)

declare %Thunk* @makeFloat(float)

declare %Thunk* @makeTuple(%Thunk*, %Thunk*, i32, i32)

declare %Thunk* @makeMaybe(%Thunk*, i32)

declare %Node* @makeNode(%Thunk*)

declare %Thunk* @makeEmptyList(i32)

declare %Thunk* @makeInfinite(i32)

declare %Thunk* @makeRangeList(%Thunk*, %Thunk*)

declare %List* @appendNode(%List*, %Node*)

declare %Thunk* @appendNodeThunk(%Thunk*, %Node*)

declare i32 @printf(i8*, ...)

declare void @printRangeList(%Thunk*)

declare void @printPrim(i8*, i32)

declare void @printAnyThunk(%Thunk*, i32*, i32)

declare void @printBool(i8)

declare void @initNativeThunks()

declare %Thunk* @mapl(%Thunk*, %Thunk*, i32)

declare %Thunk* @filterl(%Thunk*, %Thunk*, i32)
```

```
declare %Thunk* @map_listl(%Thunk*, %Thunk*, i32)

declare i32* @add(%Thunk*, %Thunk*)

declare i8* @add_eval(%Thunk*)

declare i32* @sub(%Thunk*, %Thunk*)

declare i8* @sub_eval(%Thunk*)

declare i32* @mult(%Thunk*, %Thunk*)

declare i8* @mult_eval(%Thunk*)

declare i32* @neq(%Thunk*, %Thunk*)

declare i8* @neq_eval(%Thunk*)

declare i8* @addf_eval(%Thunk*)

declare i32* @addf(%Thunk*, %Thunk*)

declare %Thunk* @init_thunk(%Thunk*, i8* (%Thunk*)*, i32, i32)

declare %Thunk* @init_thunk_literal(i8*)

declare %Thunk* @apply(%Thunk*, %Thunk*)

declare i8* @invoke(%Thunk*)

define i8* @"$eval_rec_collatz"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %thunk2 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %tload2 = load %Thunk*, %Thunk** %pthunk
  %args3 = getelementptr %Thunk, %Thunk* %tload2, i32 0, i32 3
  %loadargs4 = load %Thunk**, %Thunk*** %args3
  %args5 = getelementptr %Thunk*, %Thunk** %loadargs4, i32 1
  %loadarg6 = load %Thunk*, %Thunk** %args5
  store %Thunk* %loadarg6, %Thunk** %thunk2
  %load = load %Thunk*, %Thunk** %thunk2
  %load7 = load %Thunk*, %Thunk** %thunk1
```

```
  %result = call i8* @rec_collatz(%Thunk* %load7, %Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @"$eval_collatz"(%Thunk*) {
entry:
  %pthunk = alloca %Thunk*
  %thunk1 = alloca %Thunk*
  %ret = alloca i8*
  store %Thunk* %0, %Thunk** %pthunk
  %tload = load %Thunk*, %Thunk** %pthunk
  %args = getelementptr %Thunk, %Thunk* %tload, i32 0, i32 3
  %loadargs = load %Thunk**, %Thunk*** %args
  %args1 = getelementptr %Thunk*, %Thunk** %loadargs, i32 0
  %loadarg = load %Thunk*, %Thunk** %args1
  store %Thunk* %loadarg, %Thunk** %thunk1
  %load = load %Thunk*, %Thunk** %thunk1
  %result = call i8* @collatz(%Thunk* %load)
  store i8* %result, i8** %ret
  %retload = load i8*, i8** %ret
  ret i8* %retload
}

define i8* @rec_collatz(%Thunk*, %Thunk*) {
entry:
  %l = alloca %Thunk*
  store %Thunk* %1, %Thunk** %l
  %amanda = load %Thunk*, %Thunk** %l
  %n = alloca %Thunk*
  store %Thunk* %0, %Thunk** %n
  %amanda1 = load %Thunk*, %Thunk** %n
  %hans = alloca %Thunk*
  store %Thunk* %amanda1, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply = call %Thunk* @apply(%Thunk* @eq_init_thunk,
    %Thunk* %loaded_hans)
  %makeInt = call %Thunk* @makeInt(i32 1)
  %apply2 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %makeInt)
  %makeInt3 = call %Thunk* @makeInt(i32 1)
  %apply4 = call %Thunk* @apply(%Thunk* @cons_init_thunk,
    %Thunk* %makeInt3)
  %hans5 = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans5
  %loaded_hans6 = load %Thunk*, %Thunk** %hans5
  %apply7 = call %Thunk* @apply(%Thunk* %apply4, %Thunk* %loaded_hans6)
  %hans8 = alloca %Thunk*
  store %Thunk* %amanda1, %Thunk** %hans8
  %loaded_hans9 = load %Thunk*, %Thunk** %hans8
  %apply10 = call %Thunk* @apply(%Thunk* @mod_init_thunk,
```

```
  %Thunk* %loaded_hans9)
%makeInt11 = call %Thunk* @makeInt(i32 2)
%apply12 = call %Thunk* @apply(%Thunk* %apply10, %Thunk* %makeInt11)
%apply13 = call %Thunk* @apply(%Thunk* @eq_init_thunk,
  %Thunk* %apply12)
%makeInt14 = call %Thunk* @makeInt(i32 0)
%apply15 = call %Thunk* @apply(%Thunk* %apply13, %Thunk* %makeInt14)
%hans16 = alloca %Thunk*
store %Thunk* %amanda1, %Thunk** %hans16
%loaded_hans17 = load %Thunk*, %Thunk** %hans16
%apply18 = call %Thunk* @apply(%Thunk* @divi_init_thunk,
  %Thunk* %loaded_hans17)
%makeInt19 = call %Thunk* @makeInt(i32 2)
%apply20 = call %Thunk* @apply(%Thunk* %apply18, %Thunk* %makeInt19)
%apply21 = call %Thunk* @apply(%Thunk* @"$$rec_collatz_init_thunk",
  %Thunk* %apply20)
%hans22 = alloca %Thunk*
store %Thunk* %amanda1, %Thunk** %hans22
%loaded_hans23 = load %Thunk*, %Thunk** %hans22
%apply24 = call %Thunk* @apply(%Thunk* @cons_init_thunk,
  %Thunk* %loaded_hans23)
%hans25 = alloca %Thunk*
store %Thunk* %amanda, %Thunk** %hans25
%loaded_hans26 = load %Thunk*, %Thunk** %hans25
%apply27 = call %Thunk* @apply(%Thunk* %apply24,
  %Thunk* %loaded_hans26)
%apply28 = call %Thunk* @apply(%Thunk* %apply21, %Thunk* %apply27)
%makeInt29 = call %Thunk* @makeInt(i32 3)
%apply30 = call %Thunk* @apply(%Thunk* @mult_init_thunk,
  %Thunk* %makeInt29)
%hans31 = alloca %Thunk*
store %Thunk* %amanda1, %Thunk** %hans31
%loaded_hans32 = load %Thunk*, %Thunk** %hans31
%apply33 = call %Thunk* @apply(%Thunk* %apply30,
  %Thunk* %loaded_hans32)
%apply34 = call %Thunk* @apply(%Thunk* @add_init_thunk,
  %Thunk* %apply33)
%makeInt35 = call %Thunk* @makeInt(i32 1)
%apply36 = call %Thunk* @apply(%Thunk* %apply34, %Thunk* %makeInt35)
%apply37 = call %Thunk* @apply(%Thunk* @"$$rec_collatz_init_thunk",
  %Thunk* %apply36)
%hans38 = alloca %Thunk*
store %Thunk* %amanda1, %Thunk** %hans38
%loaded_hans39 = load %Thunk*, %Thunk** %hans38
%apply40 = call %Thunk* @apply(%Thunk* @cons_init_thunk, %Thunk*
  %loaded_hans39)
%hans41 = alloca %Thunk*
store %Thunk* %amanda, %Thunk** %hans41
%loaded_hans42 = load %Thunk*, %Thunk** %hans41
%apply43 = call %Thunk* @apply(%Thunk* %apply40, %Thunk*
  %loaded_hans42)
```

```
  %apply44 = call %Thunk* @apply(%Thunk* %apply37, %Thunk* %apply43)
  %apply45 = call %Thunk* @apply(%Thunk* @ite_init_thunk, %Thunk*
    %apply15)
  %apply46 = call %Thunk* @apply(%Thunk* %apply45, %Thunk* %apply28)
  %apply47 = call %Thunk* @apply(%Thunk* %apply46, %Thunk* %apply44)
  %apply48 = call %Thunk* @apply(%Thunk* @ite_init_thunk, %Thunk*
    %apply2)
  %apply49 = call %Thunk* @apply(%Thunk* %apply48, %Thunk* %apply7)
  %apply50 = call %Thunk* @apply(%Thunk* %apply49, %Thunk* %apply47)
  %da = call i8* @invoke(%Thunk* %apply50)
  ret i8* %da
}

define i8* @collatz(%Thunk*) {
entry:
  %x = alloca %Thunk*
  store %Thunk* %0, %Thunk** %x
  %amanda = load %Thunk*, %Thunk** %x
  %hans = alloca %Thunk*
  store %Thunk* %amanda, %Thunk** %hans
  %loaded_hans = load %Thunk*, %Thunk** %hans
  %apply = call %Thunk* @apply(%Thunk* @"$$rec_collatz_init_thunk",
    %Thunk* %loaded_hans)
  %empty = call %Thunk* @makeEmptyList(i32 0)
  %apply1 = call %Thunk* @apply(%Thunk* %apply, %Thunk* %empty)
  %da = call i8* @invoke(%Thunk* %apply1)
  ret i8* %da
}
```

7 Lessons Learned

Hollis Lehv: I learned that starting early is important. My teammates were so good about not letting bugs get us down, which made iterating more fun and less disheartening. I actually became exciting about writing and implementing functional languages.

Hans Montero: Bounce ideas off of your teammates! Sometimes it really helps to get other people's opinions of things instead of bashing your head against the wall for hours. They might think about something you missed!

Da Hua Chen: I think it would be wise to start early and work steadily throughout the semester. Have a good balance of working on your own and working with your teammates. Sometimes you'll want a challenge of writing a large part of the compiler on your own, but very often you'll want the eyes and ears of your peers. Also, pair programming is great.

Amanda Liu: It's a good idea to assign orthogonal tasks to work on in the compiler but it's equally as important to make sure that everyone is finishing

these tasks on time and that everyone is able to understand eachother's work. Otherwise it gets difficult later on to reassign tasks and work on things further down in the pipeline.

8 Appendix

8.1 scanner.mll (Amanda, Hollis, Hans, Da)

```
{ open Parser }

let letter = ['a'-'z' 'A'-'Z']
let digit = ['0'-'9']

rule token =
        parse eof               { EOF }
            (* TYPES *)
            | "int"             { INTTYPE }
            | "char"            { CHARTYPE }
            | "float"           { FLOATTYPE }
            | "bool"            { BOOLTYPE }
            | "maybe"           { MAYBE }
            (* KEYWORDS *)
            | "let"             { LET }
            | "in"              { IN }
            | "if"              { IF }
            | "then"            { THEN }
            | "else"            { ELSE }
            | "over"            { OVER }
            | "fun"             { FUN }
            (* BRACES *)
            | '['               { LBRACK }
            | ']'               { RBRACK }
            | '('               { LPAREN }
            | ')'               { RPAREN }
            (* MISC *)
            | ','               { COMMA }
            | "..."             { LRANGE }
            | "::"              { DOUBLECOL }
            | "->"              { RARROW }
            | '|'               { BAR }
            (* NUM LITERALS *)
            | digit+ as lit     { INTLIT(int_of_string lit) }
            | ((digit+ '.' digit+)) as lit { FLOATLIT(float_of_string lit)}
            (* BOOLEAN LITERALS *)
            | "true"            { TLIT }
            | "false"           { FLIT }
            (* CHAR LITERALS *)
            | "'" ( _ as c) "'" { CHARLIT(c) }
            | '\'' '\\' 'n' '\''     { CHARLIT('\n') }
            (* STRING LITERALS *)
```

```
| '"' ([^ '"']* as str) '"'  { STRLIT(str) }
(* NUM OPERATORS *)
| '+'                { PLUS }
| '-'                { MINUS }
| '/'                { DIVIDE }
| '*'                { TIMES }
| '^'                { POW }
| '%'                { MOD }
| "+."               { PLUSF }
| "-."               { MINUSF }
| "/."               { DIVIDEF }
| "*."               { TIMESF }
| "^."               { POWF }
(* BOOLEAN OPERATORS *)
| "or"               { OR }
| "and"              { AND }
| "not"              { NOT }
| "=="               { EQ }
| "==."              { EQF}
| "!="               { NEQ }
| "!=."              { NEQF }
| '<'                { LESS }
| "<."               { LESSF }
| '>'                { GREATER }
| ">."               { GREATERF }
| "<="               { LEQ }
| "<=."              { LEQF }
| ">=."              { GEQF }
| ">="               { GEQ }
(* LIST OPERATORS *)
| "cons"             { CONS }
| "head"             { HEAD }
| "tail"             { TAIL }
| "cat"              { CAT }
| "len"              { LEN }
(* TUPLE OPERATORS *)
| "first"            { FIRST }
| "sec"              { SEC }
(* MAYBE OPERATORS *)
| "none"             { NONE }
| "just"             { JUST }
| "is_none"          { IS_NONE }
| "from_just"        { FROM_JUST }
(* ASSIGN *)
| '='                { ASSIGN }
(* APPLICATION *)
| '~'                { APP }
(* CONVERSION *)
| "int_to_float"     { INT_TO_FLOAT }
(* IDENTIFIERS *)
| (letter | '_') (letter | digit | '_')* as id { IDENT(id) }
```

```
            (* WHITESPACE *)
            | [' ' '\r' '\n' '\t']   { token lexbuf }
            | "{-"                   { comment 0 lexbuf }
            | "#"                    { line_comment lexbuf }
    and line_comment =
        parse '\n'                   { token lexbuf }
            | _                      { line_comment lexbuf }
    and comment nestCount =
        parse "-}"                   { if nestCount = 0 then token lexbuf else
                                         comment (nestCount - 1) lexbuf}
            | "{-"                   { comment (nestCount + 1) lexbuf }
            | _                      { comment nestCount lexbuf }
(* STRINGS AND CHAR LITERALS
    and string_literal str =
        parse
            | "\""                   { STRLIT(str) ; token lexbuf }
    and char_literal =
            parse "\\\\"             { CHARLIT('\\') ; end_char_literal lexbuf }
            (* OHGODOHFUCK *)
            | "\\n"                  { CHARLIT('\n') ; end_char_literal lexbuf }
            | "\\t"                  { CHARLIT('\t') ; end_char_literal lexbuf }
            | "\\r"                  { CHARLIT('\r') ; end_char_literal lexbuf }
            | _ as c                 { CHARLIT(c) ; end_char_literal lexbuf }
    and end_char_literal =
        parse '\''                   { token lexbuf }
*)
```

8.2 parser.mly (Amanda, Hollis, Hans, Da)

```
%{ open Ast
   open String
   open List

   let to_char_lit c = Ast.CharLit c
   let explode s =
        let rec exp i l =
            if i < 0 then l else exp (i - 1) (s.[i] :: l) in
        exp (String.length s - 1) []
   let char_lit_list str = List.map to_char_lit (explode str)
%}

%token EOF LET IN IF THEN ELSE OVER FUN LBRACK RBRACK LPAREN RPAREN COMMA
%token LRANGE WILDCARD RARROW TLIT FLIT PLUS MINUS DIVIDE TIMES POW MOD
%token PLUSF MINUSF
%token DIVIDEF TIMESF POWF OR AND NOT EQ EQF NEQ NEQF LESS LESSF GREATER
%token GREATERF LEQ LEQF GEQ GEQF LEN CONS HEAD CAT TAIL ASSIGN BAR NEWLINE
%token DOUBLECOL INTTYPE FLOATTYPE BOOLTYPE CHARTYPE
%token INT_TO_FLOAT
%token MAYBE JUST NONE APP
%token FIRST SEC
%token IS_NONE FROM_JUST

%token <char> CHARLIT
%token <int> INTLIT
%token <string> STRLIT
%token <float> FLOATLIT
%token <string> IDENT

%left IN
%left OR AND NOT
%left EQ EQF NEQ NEQF LESS LESSF GREATER GREATERF LEQ LEQF GEQF GEQ
%right RARROW
%nonassoc MAYBE
%left ELSE

%left ASSIGN
%left PLUS MINUS PLUSF MINUSF
%left TIMES DIVIDE MOD TIMESF DIVIDEF
%nonassoc UMINUS UMINUSF
%left POW POWF
%left CONS CAT
%nonassoc FIRST SEC LEN TAIL HEAD
%nonassoc JUST IS_NONE FROM_JUST
%left APP
%left INT_TO_FLOAT /* TODO: CHECK THIS */
%nonassoc PAREN

%start program
```

```
%type <Ast.program> program

%%

program:
    | decl EOF                        { [$1] }
    | decl program             { $1 :: $2 }

decl:
    | vdef                          { $1 }
    | annotation                    { $1 }

vdef:
    | IDENT ASSIGN expr                 { Vdef($1,$3)}

ty:
    | BOOLTYPE                      { Bool }
    | INTTYPE                       { Int }
    | CHARTYPE                      { Char }
    | FLOATTYPE                     { Float }
    | LBRACK ty RBRACK              { TconList($2) }
    | LPAREN ty COMMA ty RPAREN     { TconTuple($2,$4) }
    | IDENT                         { Tvar($1) }
    | MAYBE ty                      { Tmaybe($2) }
    | ty RARROW ty                  { Tarrow($1,$3) }

annotation:
    | IDENT DOUBLECOL ty            { Annot($1,$3) }

assign:
    | IDENT ASSIGN expr              { Assign($1, $3) }

expr:

    /* SYNTACTIC EXPRESSIONS */

    | IF expr THEN expr ELSE expr { Ite($2,$4,$6) }
    | LET assign IN expr            { Let($2,$4) }
    | FUN IDENT RARROW expr        { Lambda($2,$4) }
    /* | expr expr                 { App($1,$2) } */
    | expr APP expr                    { App($1,$3) }
    | IDENT                         { Var($1) }

    | lists                           { $1 }

    /* BOOLEAN OPERATIONS */
    | expr OR expr          { App (App(Or, $1), $3) }
    | expr AND expr         { App (App(And, $1), $3) }
    | NOT expr              { App(Not, $2) }
    | expr EQ expr          { App (App(Eq, $1), $3) }
    | expr EQF expr         { App (App(EqF, $1), $3) }
```

```
| expr NEQ expr          { App (App(Neq, $1), $3) }
| expr NEQF expr         { App (App(NeqF, $1), $3) }
| expr LESS expr         { App (App(Less, $1), $3) }
| expr LESSF expr        { App (App(LessF, $1), $3) }
| expr GREATER expr      { App (App(Greater, $1), $3) }
| expr GREATERF expr     { App (App(GreaterF, $1), $3) }
| expr LEQ expr          { App (App(Leq, $1), $3) }
| expr LEQF expr         { App (App(LeqF, $1), $3) }
| expr GEQ expr          { App (App(Geq, $1), $3) }
| expr GEQF expr         { App (App(GeqF, $1), $3) }
| expr MOD expr          { App (App(Mod, $1), $3) }

/* MATH OPERATIONS */
| expr PLUS expr         { App (App(Add, $1), $3) }
| expr MINUS expr        { App (App (Sub, $1), $3) }
| expr TIMES expr        { App (App (Mult, $1), $3) }
| expr DIVIDE expr       { App (App(Div, $1), $3) }
| expr PLUSF expr        { App (App(AddF, $1), $3) }
| expr MINUSF expr       { App (App(SubF, $1), $3) }
| expr TIMESF expr       { App (App(MultF, $1), $3) }
| expr DIVIDEF expr      { App (App(DivF, $1), $3) }
| expr POW expr          { App (App(Pow, $1), $3) }
| expr POWF expr         { App (App(PowF, $1), $3) }
| MINUS expr %prec UMINUS { App(Neg, $2) }
| MINUSF expr %prec UMINUSF { App(NegF, $2) }

/* LIST OPERATIONS */
| expr CONS expr         { App (App(Cons, $1), $3) }
| HEAD expr              { App(Head, $2) }
| TAIL expr              { App (Tail, $2) }
| expr CAT expr          { App (App(Cat, $1), $3)}
| LEN expr               { App (Len, $2)}

/* LITERALS */
| literals               { $1 }

/* PARENTHESIZED EXPRESSIONS */
| LPAREN expr RPAREN %prec PAREN {$2}

/* TUPLES */
| LPAREN expr COMMA expr RPAREN { Tuple($2,$4) }
| FIRST expr                    { App(First, $2) }
| SEC expr                      { App(Sec, $2) }

/* MAYBE */
| JUST expr                     { Just($2) }
| NONE                          { None }
| IS_NONE expr                  { App(Is_none, $2) }
| FROM_JUST expr                { App(From_just, $2) }

| INT_TO_FLOAT expr             { App(Int_to_float, $2) }
```

```
    /* LIST OPERATORS */

literals:
    /* PRIMITIVE LITERALS */
    | FLIT                  { BoolLit(false) }
    | TLIT                  { BoolLit(true) }
    | CHARLIT               { CharLit($1) }
    | STRLIT                { ListLit((char_lit_list $1)) }
    | INTLIT                { IntLit($1) }
    | FLOATLIT              { FloatLit($1) }

lists:
    | LBRACK prim_list RBRACK { ListLit($2) }
    | LBRACK list_range RBRACK { $2 }
    | LBRACK list_comp RBRACK { $2 }

prim_list:
    |                       { [] }
    | expr                  { [$1] }
    | expr COMMA prim_list  { $1 :: $3 }

list_range:
    | expr LRANGE expr      { ListRange($1,$3) }

list_comp:
    | expr BAR clauses      { ListComp($1,$3) }

clauses:
    | clause                { [$1] }
    | clause COMMA clauses  { $1 :: $3 }

clause:
    | expr { Filter($1) } /*boolean filter for list comp*/
    | IDENT OVER lists { ListVBind($1,$3) } /*variable binding for list comp*/
    | IDENT OVER IDENT { ListVBind($1,Var($3)) }
```

8.3 ast.mli (Amanda)

```
type expr =
    | IntLit of int | FloatLit of float | BoolLit of bool
    | CharLit of char | WildCard
    | Add | Sub | Mult | Div | Mod | Pow
    | AddF | SubF | MultF | DivF | PowF | Neg | NegF
    | Eq | EqF | Neq | NeqF | Geq | GeqF | Leq | LeqF
    | Less | LessF | Greater | GreaterF
    | And | Or | Not
    | Cons | Cat | Len | Head | Tail
    (* Tuple operations *)
    | First | Sec
    | Tuple of (expr * expr)
    (* Maybe operations *)
```

```
    | Is_none | From_just | Just of expr | None
    | Int_to_float
    | Var of string
    | Let of (assign * expr)
    | Lambda of (string * expr)
    | App of (expr * expr)
    | Ite of (expr * expr * expr)
    | ListComp of (expr * clause list)
    | ListRange of (expr * expr)
    | ListLit of expr list
and clause =
    | ListVBind of (string * expr)
    | Filter of expr
and assign = Assign of (string * expr)

type decl =
    | Annot of (string * ty)
    | Vdef of (string * expr)
and ty = Int | Bool | Float | Char
    | Tvar of string
    | Tarrow of (ty * ty)
    | TconList of ty
    | TconTuple of (ty * ty)
    | Tforall of ((string list) * ty)
    | Tmaybe of ty

type program = decl list

type lambda_def = {
    lname: string;
    ltyp: ty;
    rtyp: ty;
    lexp: expr;
    rexp: expr;
}
```

8.4 tast.mli (Amanda)

```
open Ast

(*
type typed_expr = (typed_expr * ty)

type typed_decl = TypedVdef of (string * typed_expr)

type typed_program = typed_decl list
*)
type typed_expr = tx * ty
and
tx =
    | TIntLit of int | TFloatLit of float | TBoolLit of bool
```

```
    | TCharLit of char | TWildCard | TInt_to_float
    | TAdd | TSub | TMult | TDiv | TMod | TPow
    | TAddF | TSubF | TMultF | TDivF | TPowF | TNeg | TNegF
    | TEq | TEqF | TNeq | TNeqF | TGeq | TGeqF | TLeq | TLeqF
    | TLess | TLessF | TGreater | TGreaterF
    | TAnd | TOr | TNot
    | TCons | TCat | TLen | THead | TTail
    | TFirst | TSec
    | TTuple of (typed_expr * typed_expr)
    | TIs_none | TFrom_just | TJust of typed_expr | TNone
    | TVar of string
    | TLet of (tassign * typed_expr)
    | TLambda of (string * typed_expr)
    | TApp of (typed_expr * typed_expr)
    | TIte of (typed_expr * typed_expr * typed_expr)
    | TListComp of (typed_expr * tclause list)
    | TListRange of (typed_expr * typed_expr)
    | TListLit of typed_expr list
and tclause =
    | TListVBind of (string * typed_expr)
    | TFilter of typed_expr
and tassign = TAssign of (string * typed_expr)

type typed_decl =
    | TypedVdef of (string * typed_expr)

type tlambda_def = {
    tlname: string;
    tltyp: ty;
    trtyp: ty;
    tlexp: typed_expr;
    trexp: typed_expr;
}
```

8.5 iast.mli (Amanda, Hollis)

```
open Ast
open Miast

type inferred_expr = Ast.ty SubstMap.t * ix * ty
and
ix =
    | IIntLit of int | IFloatLit of float | IBoolLit of bool
    | ICharLit of char | IWildCard
    | IAdd | ISub | IMult | IDiv | IMod | IPow
    | IAddF | ISubF | IMultF | IDivF | IPowF | INeg | INegF
    | IEq | IEqF | INeq | INeqF | IGeq | IGeqF | ILeq | ILeqF
    | ILess | ILessF | IGreater | IGreaterF
    | IAnd | IOr | INot
    | ICons | ICat | ILen | IHead | ITail
    | IFirst | ISec
```

```
    | ITuple of (inferred_expr * inferred_expr)
    | IIs_none | IFrom_just | IJust of inferred_expr | INone
    | IInt_to_float
    | IVar of (string)
    | ILet of (Ast.ty SubstMap.t * iassign * inferred_expr)
    | ILambda of (Ast.ty SubstMap.t * string * inferred_expr)
    | IApp of (Ast.ty SubstMap.t * inferred_expr * inferred_expr)
    | IIte of (Ast.ty SubstMap.t * inferred_expr * inferred_expr * inferred_expr)
    | IListComp of (Ast.ty SubstMap.t * inferred_expr * iclause list)
    | IListRange of (Ast.ty SubstMap.t * inferred_expr * inferred_expr)
    | IListLit of (inferred_expr list)
and iclause =
    | IListVBind of (string * inferred_expr)
    | IFilter of inferred_expr
and iassign = IAssign of (string * inferred_expr)

type inferred_decl =
    | InferredVdef of (string * inferred_expr)

miast.ml (Amanda)
module SubstMap = Map.Make(String);;
```

8.6 pair_annots.ml (Hans)

```
open Ast
open Scanner
open Parser
open Get_fresh_var

let rec pair_helper decs last_annot annot main_found =
    if (List.length decs) = 0
        then if not last_annot
            then (if main_found
                then []
                else raise(Failure "main not type annotated"))
            else raise (Failure "unmatched type annotation")
    else match (List.hd decs) with
    | Annot(name, tname) -> (match last_annot with
        | true -> raise (Failure "unmatched type annotation")
        | false -> pair_helper (List.tl decs) true (name, tname)
            (if not main_found then name = "main" else main_found))
    | Vdef(vname, vexpr) -> (match last_annot with
        | true ->
            if vname = (fst annot) then
                (Annot(fst annot, snd annot), Vdef(vname, vexpr))
                :: (pair_helper (List.tl decs) false annot main_found)
            else raise (Failure "mismatched identifier name in annotation and declarat
        | false ->
            let vapair = (Annot(vname, Tvar(get_fresh "t")), Vdef(vname, vexpr)) in
            vapair :: (pair_helper (List.tl decs) false annot main_found))
let pair_av prog = match prog with
    | x :: xs -> pair_helper (x :: xs) false ("", Tvar("")) false
```

```
    | [] -> []
```

8.7 lift_lambdas.ml (Hans)

```
open Ast
open Scanner
open Parser
open Get_fresh_var
open Printf
open Pretty_type_print
module StringSet = Set.Make(String)
module StringMap = Map.Make(String)

let lamb_to_cl = Hashtbl.create 42069
let top_level_sc = ref StringSet.empty


(* Print lamb_to_cl for debugging purposes *)
let print_map _ =
    let print_closure c =
        Printf.printf " mangled name: %s; needed_params:" (fst c);
        Printf.printf " [ ";
        StringSet.iter (fun s1 -> Printf.printf "%s, " s1) (snd c);
        Printf.printf " ]\n" in
    Hashtbl.iter (fun x c -> Printf.printf "lambda: %s; \n" x; print_closure c;)
    lamb_to_cl


let rec transform_main d_list m_found = match d_list with
    | Vdef (name, exp) :: ds1 -> (if name = "main"
        then (Vdef (name,exp) :: ds1)
        else (Vdef (name,exp) :: transform_main ds1 false))
    | other :: ds2 -> other :: transform_main ds2 false
    | [] -> if m_found then [] else raise(Failure "main not found!")


let rec m_replace og_ex m_ex ex = match ex with
    | App(e1, e2) -> App(m_replace og_ex m_ex e1, m_replace og_ex m_ex e2)
    | Just(e1) -> Just(m_replace og_ex m_ex e1)
    | Tuple(e1, e2) -> Tuple(m_replace og_ex m_ex e1, m_replace og_ex m_ex e2)
    | Ite(e1, e2, e3) -> Ite(m_replace og_ex m_ex e1, m_replace og_ex m_ex e2,
    m_replace og_ex m_ex e3)
    | Lambda(e1, e2) -> Lambda(e1, m_replace og_ex m_ex e2)
    | Let(Assign(s2, e2), e3) -> Let(Assign(s2, (m_replace og_ex m_ex e2)),
    (m_replace og_ex m_ex e3))
    | ListRange(e1, e2) -> ListRange(m_replace og_ex m_ex e1,
        m_replace og_ex m_ex e2)
    | ListLit(elist) -> ListLit(List.rev (List.fold_left (fun l e -> (m_replace
        og_ex m_ex e) :: l) [] elist))
    | ListComp(e1, cl) ->
        let repl_constr = m_replace og_ex m_ex e1 in
        ListComp(repl_constr, List.rev(List.fold_left (fun l c ->
```

```
        (m_replace_clause og_ex m_ex c) :: l) [] cl))
    | other -> if other = og_ex then m_ex else ex

and m_replace_clause og_ex m_ex cl = match cl with
    | Filter(e1) -> Filter(m_replace og_ex m_ex e1)
    | ListVBind(s, e1) -> ListVBind(s, m_replace og_ex m_ex e1)


let rec add_params lam vars = match vars with
    | hd :: tl ->
        let mang_param = get_fresh ("$" ^ hd) in
        let repl_lam = m_replace (Var(hd)) (Var(mang_param)) lam in
        Lambda(mang_param, (add_params repl_lam tl))
    | [] -> lam

let rec contains n haystack = match haystack with
    | [] -> false
    | hd :: tl -> if n = hd then true else contains n tl

let rec check_clauses clauses seen_v seen_f vnames wrapped_clauses=
    match clauses with
    | [] -> if seen_v then (vnames, wrapped_clauses) else raise
    (Failure "empty clause list")
    | hd :: tl -> (match hd with
        | ListVBind(n, _) -> if seen_f then raise(Failure "unexpected variable
        binding; expecting filter") else
            if contains n vnames then raise(Failure "redeclaration of variable
            binding in clauses") else
                check_clauses tl true seen_f (n :: vnames) (hd :: wrapped_clauses)
        | Filter(e) -> if not seen_v then raise(Failure "missing variable
        binding(s)") else
            check_clauses tl seen_v true vnames ((Filter(add_params e (List.rev
            vnames))) :: wrapped_clauses))


let rec transform_comps expr = match expr with
    | Lambda(e1, e2) -> Lambda(e1, transform_comps e2)
    | App(e1, e2) -> App(transform_comps e1, transform_comps e2)
    | Just(e1) -> Just(transform_comps e1)
    | Ite(e1, e2, e3) -> Ite(transform_comps e1, transform_comps e2,
    transform_comps e3)
    | Let(Assign(n, e1), e2) -> Let(Assign(n, transform_comps e1),
    transform_comps e2)
    | ListRange(e1, e2) -> ListRange(transform_comps e1, transform_comps e2)
    | ListLit(elst) ->
        let e_list = List.rev(List.fold_left (fun l e ->
        (transform_comps e) :: l) [] elst) in
        ListLit(e_list)
    | ListComp(constr_e, cl) ->
        let trans_constr = transform_comps constr_e in
        let (c_vars, wrapped_cls) = check_clauses cl false false [] [] in
```

```
        let wrapped_constr = add_params trans_constr (List.rev c_vars) in
        ListComp((wrapped_constr), (List.rev wrapped_cls))
    | other -> other


let rec find_lambdas nested = function
    | Let(Assign(ln, Lambda(p, e8)), e9) ->
        let (lsc, lst) = get_closure_vars e8
        (StringSet.add ln (StringSet.add p !top_level_sc)) true true in
        let mangled_name = get_fresh ("$" ^ ln) in
        Hashtbl.add lamb_to_cl ln (mangled_name, lsc);
        let (_, rest_st) = if not nested then find_lambdas nested e9
                           else (lsc, WildCard
                           (* let helper construct this *) ) in

        let new_expr = Let(Assign(ln ,Lambda(p, lst)), rest_st) in

        (lsc, new_expr)

    | Let(Assign(n, e10) ,e1) ->

        let (_, st10) = find_lambdas nested e10 in
        let (_, st1) = find_lambdas nested e1 in

        (StringSet.empty, Let(Assign(n, st10), st1))


    | App(e2, e3) ->
        let (_, st2) = find_lambdas true e2 in
        let (_, st3) =  find_lambdas true e3 in

        (StringSet.empty, App(st2, st3))

    | Just(e1) -> let (_, st1) = find_lambdas true e1 in
        (StringSet.empty, Just(st1))

    | Ite(e4, e5, e6) ->
        let (_, st4) = find_lambdas nested e4 in
        let (_, st5) =  find_lambdas nested e5 in
        let (_, st6) = find_lambdas nested e6 in

        (StringSet.empty, Ite(st4, st5, st6))

    | ListRange(e1, e2) ->
        let (_, st1) = find_lambdas true e1 in
        let (_, st2) =  find_lambdas true e2 in

        (StringSet.empty, ListRange(st1, st2))

    | ListLit(e1) ->
        if List.length e1 = 0 then (StringSet.empty, ListLit([]))
```

```
            else
                let trav_list = List.fold_left (list_helperf_ex nested) [] e1 in
                (StringSet.empty, ListLit((List.rev trav_list)))

        | ListComp(e1, e2) -> (match e2 with
            | [] ->
                let (_, st1) = find_lambdas true e1 in
                (StringSet.empty, ListComp(st1, []))
            | _ ->
                let (_, st1) = find_lambdas true e1 in
                let trav_list = List.fold_left (list_helperf_cla true) [] e2 in
                (StringSet.empty, ListComp(st1, trav_list)))


        | Lambda(p2, e10) ->
            if nested then
                let (sc10, st10) = get_closure_vars e10 (StringSet.add p2
                !top_level_sc) nested true in

                let anon_name = get_fresh "$anon" in
                let new_expr = Let(Assign(anon_name, Lambda(p2, st10)),
                Var(anon_name)) in
                Hashtbl.add lamb_to_cl anon_name (anon_name, sc10);

                (sc10, new_expr)
            else
                (StringSet.empty, Lambda(p2, (snd (find_lambdas false e10))))

        | Tuple(e1, e2) ->
            let (_, st1) = find_lambdas true e1 in
            let (_, st2) =  find_lambdas true e2 in

            (StringSet.empty, Tuple(st1, st2))

        | other -> (StringSet.empty, other)

    and list_helperf_ex nested exl ex =
        let (_, st) = find_lambdas true ex in
        st :: exl

    and list_helperf_cla nested clal cla =
        let (_, st) = find_lambdas_clause true cla in
        st :: clal

    and find_lambdas_clause nested = function
        | Filter(e1) ->
            let (_, st1) = find_lambdas true e1 in
            (StringSet.empty, Filter(st1))

        | ListVBind(e1, e2) ->
            let (_, st2) =  find_lambdas true e2 in
```

```
        (StringSet.empty, ListVBind(e1, st2))

and get_closure_vars exp scope nested last_lam = match exp with
    | Let(Assign(nl, Lambda(_,_)), e1) ->
        let (il_scope, il_structure) = find_lambdas nested exp in
        let (rest_scope, rest_structure) = get_closure_vars e1
        (StringSet.add nl scope) nested false in

        let complete_il_structure =
        (match il_structure with
        | Let(Assign(nl2 ,Lambda(p, def_expr)), WildCard) -> Let(Assign(nl2,
        Lambda(p, def_expr)), rest_structure)
        | _ -> WildCard) in

        ((StringSet.union (StringSet.diff il_scope scope) rest_scope),
        complete_il_structure)

    | Let(Assign(na, e6), e1) ->
        let new_scope = (StringSet.add na scope) in
        let (sc1, st1) = (get_closure_vars e6 new_scope nested false) in
        let (sc2, st2) = (get_closure_vars e1 new_scope nested false) in
        let new_expr = Let(Assign(na, st1), st2) in

        ((StringSet.union sc1 sc2), new_expr)


    | Var(s1) ->
        let sc1 = if StringSet.mem s1 scope then StringSet.empty else
        (match (Hashtbl.find_opt lamb_to_cl s1) with
            | Some(cl) -> StringSet.add s1 (snd cl)
            | _ -> StringSet.add s1 StringSet.empty
        ) in
        (sc1, Var(s1))

    | App(e4, e5) ->
        let (sc1, st1) = (get_closure_vars e4 scope nested false) in
        let (sc2, st2) = (get_closure_vars e5 scope nested false) in

        (StringSet.union sc1 sc2, App(st1, st2))

    | Ite(e4, e5, e6) ->

        let (sc1, st1) = (get_closure_vars e4 scope nested false) in
        let (sc2, st2) = (get_closure_vars e5 scope nested false) in
        let (sc3, st3) = (get_closure_vars e6 scope nested false) in

        (StringSet.union sc1 (StringSet.union sc2 sc3), Ite(st1, st2, st3))

    | Just(e1) ->
        let (sc1, st1) = (get_closure_vars e1 scope nested false) in
```

```
            (sc1, Just(st1))

        | Tuple(e1, e2) ->
            let (sc1, st1) = (get_closure_vars e1 scope nested false) in
            let (sc2, st2) = (get_closure_vars e2 scope nested false) in

            (StringSet.union sc1 sc2, Tuple(st1, st2))

        | ListRange(e1, e2) ->
            let (sc1, st1) = (get_closure_vars e1 scope nested false) in
            let (sc2, st2) = (get_closure_vars e2 scope nested false) in

            ((StringSet.union sc1 sc2), ListRange(st1, st2))

        | ListComp(e1, e2) -> (match e2 with
            | [] ->
                let (sc1, st1) = (get_closure_vars e1 scope nested false) in

                (sc1, ListComp(st1, []))

            | cl ->
                let (sc1, st1) = (get_closure_vars e1 scope nested false) in
                let (trav_sc, trav_st) = List.fold_left (list_helperc_cl nested scope)
                (StringSet.empty, []) cl in

                ((StringSet.union sc1 trav_sc), ListComp(st1, List.rev trav_st)))


        | ListLit(e1) ->
            if List.length e1 = 0 then (StringSet.empty, ListLit([]))
            else
                let (trav_sc, trav_st) = List.fold_left (list_helperc_ex nested scope)
                (StringSet.empty, []) e1 in

                (trav_sc , ListLit((List.rev trav_st)))

        | Lambda(p, e3) ->
            if last_lam then
                let new_scope = (StringSet.add p scope) in
                let (sc1, st1) = (get_closure_vars e3 new_scope nested true) in
                (sc1, (Lambda(p, st1)))
            else
                let (il_scope, il_structure) = find_lambdas nested exp in
                let set_diff = (StringSet.diff il_scope scope) in
                (set_diff, il_structure)

        | other -> (StringSet.empty, other)


    and list_helperc_ex nested scope exl ex =
        let (sc, st) = get_closure_vars ex scope nested false in
```

```
    ((StringSet.union sc (fst exl), st :: (snd exl)))

and list_helperc_cl nested scope clal cl =
    let (sc, st) = get_closure_vars_clause cl scope nested in

    ((StringSet.union sc (fst clal)), st :: (snd clal))

and get_closure_vars_clause exp scope nested = match exp with
    | Filter(e1) ->
        let (sc1, st1) = get_closure_vars e1 scope nested false in

        (sc1, Filter(st1))

    | ListVBind(e1, e2) ->
        let (sc2, st2) =  find_lambdas true e2 in

        (sc2, ListVBind(e1, st2))

let rec wrap_app la vars = match vars with
    | hd :: tl ->
        let app1 = App(la, Var(hd)) in
        wrap_app app1 tl
    | [] -> la

let rec wrap_lambda lam cl cl_to_mang = match cl with
    | hd :: tl -> let var = (match StringMap.find_opt hd cl_to_mang with
                            | Some v -> v
                            | None -> raise (Failure (hd^" not found"))) in
        Lambda(var, (wrap_lambda lam tl cl_to_mang))
    | [] -> lam

let rec repl_body body cl_to_mang tl_seen = match body with
    | Var(s1) -> (match Hashtbl.find_opt lamb_to_cl s1 with
        | Some(lc) -> (* oh god oh FUCK it's a lambda call.. let's close it rq *)
            let closure_vars = StringSet.elements (snd lc) in
            let params_to_wrap = List.rev(List.fold_left (fun pl p ->
                match (StringMap.find_opt p cl_to_mang) with
                    | Some(cp) -> cp :: pl
                    | _ -> p :: pl) [] closure_vars) in
            let new_name = (*check if this lambda came from closure *)
            (match StringMap.find_opt s1 cl_to_mang with
                | Some(m) -> m
                | _ -> s1) in
            let wrapped_var = wrap_app (Var(new_name)) params_to_wrap in
            wrapped_var

        | _ -> ( (* not a lambda, let's check if it's a closure variable *)
            match StringMap.find_opt s1 cl_to_mang with
            | Some(mp) -> Var(mp)
            (* ok epic, it's a closure variable..
```

```
            let's use the mangled parameter name *)
            | _ -> Var(s1))) (* not from closure, just use the same name*)
    | App(e1, e2) -> App(repl_body e1 cl_to_mang tl_seen,
    repl_body e2 cl_to_mang tl_seen)
    | Just(e1) -> Just(repl_body e1 cl_to_mang tl_seen)
    | Tuple(e1, e2) -> Tuple(repl_body e1 cl_to_mang tl_seen, repl_body e2
    cl_to_mang tl_seen)
    | Let(Assign(n, e1), e2) -> (match e1 with
        | Lambda(_, _) -> Let(Assign(n, e1), (if not tl_seen then (repl_body e2
        cl_to_mang tl_seen) else e2))
        | _ -> Let(Assign(n, repl_body e1 cl_to_mang tl_seen), repl_body e2
        cl_to_mang tl_seen))
    | Ite(e1, e2, e3) -> Ite(repl_body e1 cl_to_mang tl_seen, repl_body e2
    cl_to_mang tl_seen, repl_body e3 cl_to_mang tl_seen)
    | Lambda(n, e1) -> Lambda(n, repl_body e1 cl_to_mang tl_seen)
    | ListRange(e1, e2) -> ListRange(repl_body e1 cl_to_mang tl_seen,
    repl_body e2 cl_to_mang tl_seen)
    | ListLit(el) ->
        let rl = List.rev(List.fold_left (fun lst le -> (repl_body le
        cl_to_mang tl_seen) :: lst) [] el) in
        ListLit(rl)
    | ListComp(e1, cl) ->
        let rcl = List.rev(List.fold_left (fun lst lc ->
        (repl_bodyc lc cl_to_mang tl_seen) :: lst) [] cl) in
        ListComp(repl_body e1 cl_to_mang tl_seen, rcl)
    | other -> other

and repl_bodyc body cl_to_mang tl_seen = match body with
    | ListVBind(s, e1) -> ListVBind(s, repl_body e1 cl_to_mang tl_seen)
    | Filter(e1) -> Filter(repl_body e1 cl_to_mang tl_seen)

let rec mangle_close e nested tl_seen = match e with
    | Var(s1) -> Var(s1)
    | App(e1, e2) -> App(mangle_close e1 nested tl_seen,
    mangle_close e2 nested tl_seen)
    | Just(e1) -> Just(mangle_close e1 nested tl_seen)
    | Tuple(e1, e2) -> Tuple(mangle_close e1 nested tl_seen,
    mangle_close e2 nested tl_seen)
    | Ite(e3, e4, e5) -> Ite(mangle_close e3 nested tl_seen,
    mangle_close e4 nested tl_seen, mangle_close e5 nested tl_seen)
    | Let(Assign(n, rexp), inexp) -> (match rexp with
        | Lambda(lparam, lbody) ->
            let mc_lbody = mangle_close lbody true false in
            let closure_info = (match Hashtbl.find_opt lamb_to_cl n with
                    | Some r -> r
                    | None -> raise (Failure ("couldn't find " ^ n))) in
            let cl_vars = StringSet.elements (snd closure_info) in
            let clv_to_mang = List.fold_left (fun mp c ->
            StringMap.add c (get_fresh ("$" ^ c)) mp) StringMap.empty cl_vars in
            let mang_body = repl_body mc_lbody clv_to_mang tl_seen in
            let w_lambda = wrap_lambda (Lambda(lparam, mang_body))
```

```
            cl_vars clv_to_mang in
            let man_in = mangle_close inexp nested true in
            let man2_in_expr = if (not (nested || tl_seen))
                then (repl_body man_in StringMap.empty tl_seen)
                else man_in in
            Let(Assign(n, w_lambda), man2_in_expr)
        | other -> Let(Assign(n, mangle_close other nested false),
        (mangle_close inexp nested tl_seen)))
    | Lambda(n, e1) -> Lambda(n, mangle_close e1 nested tl_seen)
    | ListRange(e1, e2) -> ListRange(mangle_close e1 nested tl_seen,
    mangle_close e2 nested tl_seen)
    | ListLit(el) ->
        let ml = List.rev(List.fold_left (fun lst le ->
        (mangle_close le nested tl_seen) :: lst) [] el) in
        ListLit(ml)
    | ListComp(e1, cl) ->
        let ml = List.rev(List.fold_left (fun lst lc ->
        (mangle_closec lc nested tl_seen) :: lst ) [] cl) in
        ListComp(mangle_close e1 nested tl_seen, ml)
    | other -> other

and mangle_closec c nested tl_seen = match c with
    | ListVBind(s, e1) -> ListVBind(s, mangle_close e1 nested tl_seen)
    | Filter(e1) -> Filter(mangle_close e1 nested tl_seen)


let rec lift exp decl_list = match exp with
    | Let(Assign(ln, Lambda(p, e8)), e9) ->
        let (new_lbody, new_dlist) = lift e8 decl_list in
        let lifted_l = Vdef(ln, Lambda(p, new_lbody)) in
        lift e9 (new_dlist @ [lifted_l])
    | App(e1, e2) ->
        let (body1, dlist1) = lift e1 decl_list in
        let (body2, dlist2) = lift e2 dlist1 in
        (App(body1, body2), dlist2)
    | Just(e1) ->
        let (body1, dlist1) = lift e1 decl_list in
        (Just(body1), dlist1)
    | Tuple(e1, e2) ->
        let (body1, dlist1) = lift e1 decl_list in
        let (body2, dlist2) = lift e2 dlist1 in
        (Tuple(body1, body2), dlist2)
    | Ite(e1, e2, e3) ->
        let (body1, dlist1) = lift e1 decl_list in
        let (body2, dlist2) = lift e2 dlist1 in
        let (body3, dlist3) = lift e3 dlist2 in
        (Ite(body1, body2, body3), dlist3)
    | Let(Assign(n, e1), e2) ->
        let (body1, dlist1) = lift e1 decl_list in
        let (body2, dlist2) = lift e2 dlist1 in
        (Let(Assign(n, body1), body2), dlist2)
```

```
    | ListRange(e1, e2) ->
        let (body1, dlist1) = lift e1 decl_list in
        let (body2, dlist2) = lift e2 dlist1 in
        (ListRange(body1, body2), dlist2)
    | ListLit(elist) ->
        let (blist, dlist) = List.fold_left lift_helpere ([], decl_list) elist in
        (ListLit(List.rev blist), dlist)
    | ListComp(e1, cl) ->
        let (body1, dlist1) = lift e1 decl_list in
        let(clist, dlist2) = List.fold_left lift_helperc ([], dlist1) cl in
        (ListComp(body1, clist), dlist2)
    | Var(s1) ->
        (Var(s1), decl_list)
    | Lambda(e1, e2) ->
        let (body2, dlist2) = lift e2 decl_list in
        (Lambda(e1, body2), dlist2)
    | other -> (other, decl_list)

and lift_helpere dl e =
    let (body1, dlist1) = lift e (snd dl) in
    ((body1 :: (fst dl)), dlist1)

and lift_helperc cl c = match c with
    | ListVBind(e1, e2) ->
        let (body2, dlist2) = lift e2 (snd cl) in
        ((ListVBind(e1, body2) :: (fst cl)), dlist2)
    | Filter(e1) ->
        let (body1, dlist1) = lift e1 (snd cl) in
        ((Filter(body1) :: (fst cl)), dlist1)

let rec get_last lst = match lst with
    | [el] -> el
    | hd :: tl -> get_last tl
    | _ -> raise(Failure "can't get last of empty list")

let rec rest_list lst = match lst with
    | [el] -> []
    | hd :: tl -> (hd :: (rest_list tl))
    | _ -> []

let lift_decl curr_list d = match d with
    | Vdef(n, e) ->
            let wraplc_ast = transform_comps e in
            (*print_endline ("+++transformed comp++\n" ^
            (ast_to_str wraplc_ast) ^ "\n++++++++++");*)
            let (_, nl_ast) = find_lambdas false wraplc_ast in ();
            (*print_endline ("-------findlam------\n" ^
            (ast_to_str nl_ast) ^ "\n--------");*)
            let mang_ast = (*print_map 0;*) mangle_close nl_ast false false in
            (*print_endline ("+++++++++mangled+++++++\n" ^
            (ast_to_str mang_ast) ^ "\n++++++++++");*)
```

```
            let (corpse, l_decs) = lift mang_ast [] in
            (*print_map 0;*)
            if List.length curr_list = 0 then (l_decs @ [Vdef(n, corpse)]) else (
                let last_elem = get_last curr_list in
                let rest_list = rest_list curr_list in
                (match last_elem with
                    | Annot(an, _) ->
                        if an = n then (rest_list @ l_decs @ [last_elem] @
                        [Vdef(n, corpse)])
                        else (curr_list @ l_decs @ [Vdef(n, corpse)])
                    | _ -> curr_list @ l_decs @ [Vdef(n, corpse)]))

    | annot -> curr_list @ [annot]

let get_top_sc plist =
    top_level_sc := List.fold_left (fun s d -> match d with
        | Vdef(n, e) -> StringSet.add n s
        | _ -> s) StringSet.empty plist

let close_and_lift ast =
    get_top_sc ast;
    List.fold_left lift_decl [] ast
```

8.8 check_main.ml (Hollis)

```
open Ast
open Scanner
open Parser

let rec find_main prog =
        match prog with
            | x :: xs -> (
                match x with
                    | Annot(_,_) -> find_main xs
                    | Vdef(ident,_) -> if ident = "main" then "YAY" else
                            find_main xs
            )
            | [] -> "NAY"
```

8.9 check_lists.ml (Hollis)

```
open Ast
open Scanner
open Parser

let rec check_list_clauses clauses =
    match clauses with
        | hd :: tl -> (
                match hd with
                    | ListVBind(_, lst) -> "GOOD LIST COMPREHENSION"
                    | _ -> "BAD LIST COMPREHENSION"
                )
```

```
        | [] -> ""


let rec check_list_comps prog =
    match prog with
        | x :: xs -> ( match x with
                | Vdef(ident, expr) -> (
                    match expr with
                        | ListComp(expr, lst) -> check_list_clauses lst
                        | _ -> check_list_comps xs
                    )
                | Annot(_,_) -> check_list_comps xs
        )
        | []-> ""
```

8.10 remove_substs.ml (Amanda)

```
open Iast
open Tast
open Type_inference

let rec remove_subst_expr subst = function
    | (_,IIntLit i,t) -> (TIntLit i,t)
    | (_,IFloatLit f,t) -> (TFloatLit f, t)
    | (_,IBoolLit b,t) -> (TBoolLit b,t)
    | (_,ICharLit c,t) -> (TCharLit c,t)
    | (_,ITuple(ix1,ix2),t) ->
      (TTuple(remove_subst_expr subst ix1, remove_subst_expr subst ix2),
      apply subst t)
    | (_,IWildCard,t) -> (TWildCard,apply subst t) | (_,IAdd,t) -> (TAdd,t)
    | (_,ISub,t) -> (TSub,t) | (_,IMult,t) -> (TMult,t)
    | (_,IDiv,t) -> (TDiv,t) | (_,IMod,t) -> (TMod,t) | (_,IPow,t) ->
      (TPow,t)
    | (_,IAddF,t) -> (TAddF,t) | (_,ISubF,t) -> (TSubF,t)
    | (_,IMultF,t) -> (TMultF,t) | (_,IDivF,t) -> (TDivF,t)
    | (_,IPowF,t) -> (TPowF,t) | (_,INegF,t) -> (TNegF,t)
    | (_,INeg,t) -> (TNeg,t) | (_,IEq,t) -> (TEq,t) | (_,IEqF,t) -> (TEqF,t)
    | (_,INeq,t) -> (TNeq,t) | (_,INeqF,t) -> (TNeqF,t)
    | (_,IGeq,t) -> (TGeq,t) | (_,IGeqF,t) -> (TGeqF,t)
    | (_,ILeq,t) -> (TLeq,t) | (_,ILeqF,t) -> (TLeqF,t)
    | (_,ILess,t) -> (TLess,t) | (_,ILessF,t) -> (TLessF,t)
    | (_,IGreater,t) -> (TGreater,t) | (_,IGreaterF,t) -> (TGreaterF,t)
    | (_,IAnd,t) -> (TAnd,t) | (_,IOr,t) -> (TOr,t) | (_,INot,t) -> (TNot,t)
    | (_,ICons,t) -> (TCons,apply subst t) | (_,ICat,t) ->
      (TCat,apply subst t)
    | (_,ILen,t) -> (TLen,apply subst t) | (_,IHead,t) ->
      (THead,apply subst t)
    | (_,ITail,t) -> (TTail,apply subst t)
    | (_,IInt_to_float,t) -> (TInt_to_float,t)
    | (_,IVar (str),t) -> (TVar str,apply subst t)
    | (_,ILet (_, iassign, ix1),t) ->
```

```
        (TLet(remove_subst_iassign subst iassign,
            remove_subst_expr subst ix1),apply subst t)
    | (_,ILambda(_, var, ix2),t) ->
        (TLambda(var, remove_subst_expr subst ix2),apply subst t)
    | (_,IApp(_, ix1, ix2),t) ->
        (TApp(remove_subst_expr subst ix1, remove_subst_expr
            subst ix2),apply subst t)
    | (_,IIte(_,ix1, ix2, ix3),t) ->
        (TIte(remove_subst_expr subst ix1, remove_subst_expr subst ix2,
            remove_subst_expr subst ix3),apply subst t)
    | (_,IListComp(_,ix1, iclause_list),t) ->
        (TListComp(remove_subst_expr subst ix1,
        List.map (remove_subst_clause subst) iclause_list),apply subst t)
    | (_,IListRange(_,ix1,ix2),t) ->
        (TListRange(remove_subst_expr subst ix1, remove_subst_expr
            subst ix2),apply subst t)
    | (_,(IListLit(ix_list)),t) ->
        (TListLit (List.map (remove_subst_expr subst) ix_list),apply subst t)
    | (_,(IFirst),t) -> (TFirst,apply subst t)
    | (_,(ISec),t) -> (TSec,apply subst t)
    | (_,(IJust ix),t) -> (TJust(remove_subst_expr subst ix),apply subst t)
    | (_,(INone),t) -> (TNone,apply subst t)
    | (_,(IIs_none),t) -> (TIs_none,apply subst t)
    | (_,IFrom_just,t) -> (TFrom_just,apply subst t)
and remove_subst_clause subst = function
    | IListVBind(str,ix2) ->
        TListVBind(str, remove_subst_expr subst ix2)
    | IFilter ix1 ->
        TFilter (remove_subst_expr subst ix1)
and remove_subst_iassign subst = function
    | IAssign(str,ix2) ->
        TAssign(str, remove_subst_expr subst ix2)

let remove_subst subst = function
    |InferredVdef(name,expr) -> TypedVdef(name, remove_subst_expr
        subst expr)

let rec remove_subst_pairs subst = function
    | [] -> []
    | (annot, infvdef)::xs ->
        (annot, remove_subst subst infvdef)::(remove_subst_pairs
            subst xs)
```

8.11 type_inference.ml (Amanda, Hollis)

```
open Ast
open Tast
open Get_fresh_var
open Iast
open List
open Pretty_type_print
```

```
module SS = Set.Make(String);;
module SMap = Map.Make(String);;
module SubstMap = Map.Make(String);;

(* mappings from term variables to tforall *)
module TyEnvMap = Map.Make(String);;

(* returns a set of free type variables *)

let printEnv env =
    print_string "[";
    TyEnvMap.iter (fun key -> fun ty ->
        print_string (key ^ " :: " ^ (ty_to_str ty)^", ")) env;
    print_endline "]"


let rec ftv = function
    | Tvar(n) -> SS.add n SS.empty
    | Int -> SS.empty
    | Bool -> SS.empty
    | Float -> SS.empty
    | Char -> SS.empty
    | Tarrow (t1, t2) -> SS.union (ftv t1) (ftv t2)
    | TconList (t) -> ftv t
    | TconTuple (t1, t2) -> SS.union (ftv t1) (ftv t2)
    | Tforall (stlst, t) -> SS.diff (ftv t) (SS.of_list stlst)
    | Tmaybe (t) -> ftv t

let rec apply s = function
    | Tvar(n) ->
        (match SubstMap.find_opt n s with
        | Some t -> apply s t
        | None -> Tvar(n)
    )
    | Tarrow (t1, t2) -> Tarrow ( apply  s t1, apply s t2 )
    | TconList (t) -> TconList (apply s t)
    | TconTuple (t1, t2) -> TconTuple (apply s t1, apply s t2)
    | Tforall (stlst, t) -> Tforall (stlst,
        apply (List.fold_right SubstMap.remove stlst s) t)
    | Tmaybe(t) -> Tmaybe(apply s t)
    | t -> t

let collision key e1 e2 = Some e1

let nullSubst : ty SubstMap.t = SubstMap.empty

let composeSubst (s1 : ty SubstMap.t) (s2 : ty SubstMap.t) =
    SubstMap.union collision (SubstMap.map (apply s1) s2)
        (SubstMap.map (apply s2) s1)

(* removes element from typing environment *)
```

```
let remove (env : ty SubstMap.t) var =
    SubstMap.remove var env

let getElem = function
    | (key, a) -> a

let getElems mp = List.map getElem (TyEnvMap.bindings mp)

let printSubst s = print_string "{" ;
                   SubstMap.iter
                   (fun key -> fun ty ->
                   print_string (key ^ ": " ^ (ty_to_str ty)^", ")) s;
                   print_endline "}"

(*  get elements of the map (not the keys),
    and map ftv over them then make a new set with those ftvs*)
let ftvenv env =
    (List.fold_right ( SS.union ) (List.map ftv (getElems env)) SS.empty )

let applyenv subst env = (TyEnvMap.map (apply subst) env)

let generalize env t =
    let vars = SS.elements (SS.diff (ftv t) (ftvenv env)) in
        Tforall(vars, t)

let newTyVar prefix =
    let str = get_fresh prefix in Tvar(str)

let rec zip lst1 lst2 = match lst1, lst2 with
    | [], _ -> []
    | _, [] -> []
    | (x :: xs), (y :: ys) -> (x, y) :: (zip xs ys)

let rec unzip = function
    | (a,b)::xs -> let rest = unzip xs in
        let resta = fst rest in
        let restb = snd rest in
        (a::resta,b::restb)
    | [] -> ([],[])

let rec map_from_list = function
    | [] -> SubstMap.empty
    | (t1, t2) :: tl -> SubstMap.add t1 t2 (map_from_list tl)

let instantiate = function
    | Tforall(vars, t) ->
        let nvars = List.map (fun var -> newTyVar(var)) vars in
        let s = map_from_list (zip vars nvars) in
        apply s t
    | t -> t
```

```
let varBind u t = match u, t with
    | u, Tvar(x) -> if (String.equal u x)
                    then nullSubst
                    else SubstMap.add u (Tvar(x)) SubstMap.empty
    | u, t when SS.mem u (ftv t) -> raise
        (Failure ("Cannot bind "^u^" to "^(ty_to_str t)) )
    | _,_ -> SubstMap.add u t SubstMap.empty

let rec mgu ty1 ty2 =
    match ty1, ty2 with
    | Tarrow(l, r), Tarrow(l', r') ->
            let s1 = mgu l l' in
            let s2 = mgu (apply s1 r) (apply s1 r') in
            composeSubst s1 s2
    | Tvar(u), t -> varBind u t
    | t, Tvar(u) -> varBind u t
    | Int, Int -> nullSubst
    | Bool, Bool -> nullSubst
    | Float, Float -> nullSubst
    | Char, Char -> nullSubst
    | TconList(t), TconList(t') -> mgu t t'
    | Tmaybe(t), Tmaybe(t') -> mgu t t'
    | TconTuple(l, r), TconTuple(l', r') ->
            let s1 = mgu l l' in
            let s2 = mgu (apply s1 r) (apply s1 r') in
            composeSubst s1 s2
    | t1, t2 -> raise(Failure ((ty_to_str ty1) ^ " types do not unify " ^
(ty_to_str ty2)))

(* Collects tvars in a list; doesn't work for tforalls because we
 * shouldn't need to call it on a tforall *)
let collect_tvar =
    let rec collect genlist = function
        | (Tvar var) -> var::genlist
        | (TconList ty) -> (collect genlist ty)
        | (TconTuple (t1,t2)) -> let l1 = collect genlist t1 in
        let l2 = collect l1 t2 in l2
        | (Tarrow (t1,t2)) -> let l1 = collect genlist t1 in
        let l2 = collect l1 t2 in l2
        | (Tmaybe ty) -> (collect genlist ty)
        | (Tforall _) -> raise (Failure "can't generalize tforalls")
        | _ -> genlist
        in collect []

        (* Returns tforall if there are tvars, normal type if not *)
let simple_generalize ty =
    let gen_list = collect_tvar ty in
    if (List.length gen_list) = 0 then ty else (Tforall (gen_list,ty))

(* Takes an AST and returns a TAST (typed AST) *)
let rec ti env expr =
```

```
let rec ti_vbinds env = function
    | ((ListVBind(v,e))::xs) ->
        let (s,ix,ty) as ixpr =
        (match e with
        | (ListComp _) as l -> ti env l
        | (ListRange _) as l -> ti env l
        | (ListLit _) as l -> ti env l
        | (Var _) as l -> ti env l
        | t -> raise (Failure(
        "list comprehension variable "^v^" is not defined over list "
        ^(ast_to_str t))))
        in (match ty with
            | TconList _ -> (IListVBind(v,ixpr))::(ti_vbinds
                (applyenv s env) xs)
            | Tvar t -> (IListVBind(v,ixpr))::(ti_vbinds
                (applyenv s env) xs)
            | _ ->  raise (Failure(
        "list comprehension variable "^v^" is not defined over list")))

    | [] -> []
    | _ -> raise (Failure "Unexpected filter")
in
let rec ti_filters env = function
    | ((Filter e)::xs) ->
        let (s,ix,ty) as ixpr = ti env e in
        (IFilter ixpr)::(ti_filters (applyenv s env) xs)
    | [] -> []
    | _ -> raise (Failure "Unexpected list vbind")
in
let collect_vbinds_filters mixed_list =
    let rec collect l tuple =
        match tuple with (vbinds,filters) ->
        (match l with
        | (ListVBind(t)::xs) -> collect xs ((ListVBind(t))::vbinds,filters)
        | (Filter(t) ::xs) -> collect xs (vbinds,(Filter(t))::filters)
        | [] -> (List.rev vbinds, List.rev filters))
    in collect mixed_list ([],[])
in
let rec tys_from_vbinds = function
    | (IListVBind(_,(_,_,(TconList ty))))::xs ->
        (ty,nullSubst)::(tys_from_vbinds xs)
    | (IListVBind(_,(_,_,(Tvar ty))))::xs ->
        let oop = newTyVar "oop"
        in (oop, mgu (TconList oop) (Tvar ty))::(tys_from_vbinds xs)
    | [] -> []
    | _ -> raise
        (Failure "list comprehension variable is not defined over list")
in
let rec tys_from_filters = function
    | (IFilter(_,_,ty))::xs -> ty::(tys_from_filters xs)
    | [] -> []
```

```
    | _ -> raise (Failure "error in tys_from_filters")
in
let rec substs_from_vbinds = function
    | (IListVBind(_,(s,_,_)))::xs -> composeSubst s
        (substs_from_vbinds xs)
    | [] -> nullSubst
    | _ -> raise
        (Failure "list comprehension variable is not defined over list")
in
let rec substs_from_filters = function
    | (IFilter(s,_,_))::xs -> composeSubst s (substs_from_filters xs)
    | [] -> nullSubst
    | _ -> raise (Failure "error in tys_from_filters")

in
let rec merge_tys_filter tlist filter_ty = match tlist with
    | (ty::xs) -> (match filter_ty with
        | Tarrow(arg,ret) ->
            let s1 = mgu arg ty in
            let s2 = merge_tys_filter xs ret in
            composeSubst s1 s2
        | _ -> raise (Failure "improper filter
            type in list comprehension"))
    | [] -> (mgu filter_ty Bool)
in
let rec merge_tys_expr tlist expr_ty = match tlist with
    |(ty::xs) -> (match expr_ty with
        | Tarrow(arg,ret) ->
            let s1 = mgu arg ty in
            let (s2,ret_ty) = merge_tys_expr xs ret in
            let s3 = composeSubst s1 s2 in
            (s3, apply s3 ret_ty)
        | _ -> raise (Failure "improper variable bindings in list comp"))
    | [] -> (nullSubst,expr_ty)
in
let type_listcomp env comp = match comp with
    (ListComp(e,clauses)) ->
    let (s,ix,ty) as ixpr = ti env e in
    let (vbinds,filters) = collect_vbinds_filters clauses in
    let (ivbinds,ifilters) =(ti_vbinds env vbinds,ti_filters
        env filters) in
    let (vbind_tys,vbind_substs) = unzip (tys_from_vbinds ivbinds) in
    let polyvbind_substs =  List.fold_left
        (fun s1 -> fun s2 -> composeSubst s1 s2)
        (List.hd vbind_substs) vbind_substs in
    let vsubsts = composeSubst polyvbind_substs
        (substs_from_vbinds ivbinds) in
    let fsubsts = substs_from_filters ifilters in
    let filter_tys = tys_from_filters ifilters in
    let filtsubst = composeSubst fsubsts (List.fold_left
        (fun su -> fun t -> composeSubst (merge_tys_filter
```

```
            vbind_tys t) su)
        nullSubst filter_tys) in
    let (esubst,ety) = merge_tys_expr vbind_tys ty in
    let allsubst = composeSubst vsubsts (composeSubst filtsubst
        esubst) in
    let ret_ty = apply allsubst ety in
    (allsubst,IListComp(allsubst,ixpr,(ivbinds@ifilters)),
    TconList(ret_ty))
    | _ -> raise (Failure ("List comp expected"))
in
(*************************EXPRS*****************************)
match expr with
| IntLit i -> (nullSubst, IIntLit i, Int)
| FloatLit f -> (nullSubst, IFloatLit f,Float)
| CharLit c -> (nullSubst, ICharLit c,Char)
| BoolLit b -> (nullSubst, IBoolLit b,Bool)
| Tuple (e1,e2) ->
    let (s1,tex1,ty1) as ix1 = ti env e1 in
    let (s2,tex2,ty2) as ix2 = ti (applyenv s1 env) e2 in
    let s3 = composeSubst s1 s2 in
    (s3
    , ITuple(ix1,ix2)
    , TconTuple(apply s3 ty1, apply s3 ty2))
| ListLit [] -> (nullSubst, IListLit [], TconList(newTyVar "a"))
| ListLit l -> let iexpr_list = List.map (ti env) l in
    (match iexpr_list with
    (* collect all substs; apply substs on elements and final type *)
    | ix_list ->
        let fullSubst =
        fold_left (fun s1 (s2,_,_) -> composeSubst s1 s2) env ix_list in
        let merged_ix_list = List.map
            (fun (env,e,t) -> (env,e, apply fullSubst t)) ix_list in
        let (_,_,ty) = List.hd merged_ix_list in
        (fullSubst, IListLit(merged_ix_list), TconList ty))
| ListRange(e1, e2) ->
    let (subst1, tex1, ty1) = ti env e1 in
    let (subst2,tex2, ty2) = ti (applyenv subst1 env) e2 in
    let subst3 = mgu (apply subst2 ty1) ty2 in
    let subst4 = mgu (apply subst3 ty2) Int in
    let fullsubst = composeSubst subst1 (composeSubst subst2
        (composeSubst subst3 subst4)) in
    (fullsubst
        , IListRange(subst4,
            (subst1,tex1,apply fullsubst ty1),
            (subst2,tex2,apply fullsubst ty2))
        , TconList Int)
| None -> let polyty = newTyVar "a" in
    (nullSubst, INone, Tmaybe polyty)
| Just e -> let (s,ix,t) as ixpr = ti env e in
    (s, IJust ixpr, Tmaybe t)
    | ListComp(_) as comp -> type_listcomp env comp
```

```
    | Var n -> let sigma = TyEnvMap.find_opt n env in
            (match sigma with
            | None -> raise(Failure("unbound variable " ^ n))
            | Some si -> let t = instantiate si in
                (nullSubst, IVar n, t)
            )
    | Let(Assign(x, e1), e2) ->
            let (s1,tex1,t1) as ix1 = ti env e1 in
            let t' = generalize (applyenv s1 env) t1 in
            let env'' = (TyEnvMap.add x t' (applyenv s1 env)) in
            let (s2, tex2, t2) as ix2 = ti (applyenv s1 env'') e2 in
            (composeSubst s1 s2
            , ILet(composeSubst s1 s2, IAssign(x, ix1), ix2)
            , t2)
| Lambda( n, e ) ->
    let tv = newTyVar n in
    let env' = remove env n in
    let env'' = SubstMap.union collision env'
        (SubstMap.singleton n (Tforall([], tv)) ) in
    let (s1, tex1, t1) as ix1 = ti env'' e in
    (s1, ILambda (s1, n, ix1), Tarrow( (apply s1 tv), t1 ))
| App(e1,e2) ->
    let tv = newTyVar "app" in
    let (s1, tx1, t1) as ix1 = ti env e1 in
    let (s2, tx2, t2) as ix2 = ti (applyenv s1 env) e2 in
    let s3 = mgu (apply s2 t1) (Tarrow( t2, tv)) in
    ((composeSubst (composeSubst s1 s2) s3)
    , IApp(s3,ix1,ix2)
    , apply s3 tv)
| Ite(e1,e2,e3) ->
    let (s1,tx1,t1) as ix1 = ti env e1 in
    (*first expr must be boolean*)
    let boolSubst = composeSubst (mgu Bool t1) s1 in
    let (s2,tx2,t2) as ix2 = ti (applyenv boolSubst env) e2 in
    let s' = composeSubst boolSubst s2 in
    let (s3,tx2,t3) as ix3 = ti (applyenv s' env) e3 in
    let s'' = mgu t2 t3 in
    let fullSubst = composeSubst s' s'' in
    (fullSubst
    , IIte(fullSubst, ix1,ix2,ix3)
    , apply fullSubst t2)
| Add -> (nullSubst, IAdd, Tarrow(Int, Tarrow(Int,Int)))
    | Sub -> (nullSubst, ISub, Tarrow(Int, Tarrow(Int,Int)))
    | Mult -> (nullSubst, IMult, Tarrow(Int, Tarrow(Int,Int)))
    | Div -> (nullSubst, IDiv, Tarrow(Int, Tarrow(Int,Int)))
    | Mod -> (nullSubst, IMod, Tarrow(Int,Tarrow(Int,Int)))
    | Pow -> (nullSubst, IPow, Tarrow(Int,Tarrow(Int,Int)))
    | AddF -> (nullSubst, IAddF, Tarrow(Float, Tarrow(Float,Float)))
    | SubF -> (nullSubst, ISubF, Tarrow(Float, Tarrow(Float,Float)))
    | MultF -> (nullSubst, IMultF, Tarrow(Float, Tarrow(Float,Float)))
    | DivF -> (nullSubst, IDivF, Tarrow(Float,Tarrow(Float,Float)))
```

```
            | PowF -> (nullSubst, IPowF, Tarrow(Float,Tarrow(Float,Float)))
            | Neg -> (nullSubst, INeg, Tarrow(Int, Int))
            | NegF -> (nullSubst, INegF, Tarrow(Float, Float))
            | Eq -> (nullSubst, IEq, Tarrow(Int,Tarrow(Int,Bool)))
            | EqF -> (nullSubst, IEqF, Tarrow(Float,Tarrow(Float,Bool)))
            | Neq -> (nullSubst, INeq, Tarrow(Int,Tarrow(Int,Bool)))
            | NeqF -> (nullSubst, INeqF, Tarrow(Float,Tarrow(Float,Bool)))
            | Geq -> (nullSubst, IGeq, Tarrow(Int,Tarrow(Int,Bool)))
            | GeqF -> (nullSubst, IGeqF, Tarrow(Float,Tarrow(Float,Bool)))
            | Leq -> (nullSubst, ILeq, Tarrow(Int,Tarrow(Int,Bool)))
            | LeqF -> (nullSubst, ILeqF, Tarrow(Float,Tarrow(Float,Bool)))
            | Less -> (nullSubst, ILess, Tarrow(Int,Tarrow(Int,Bool)))
            | LessF -> (nullSubst, ILessF, Tarrow(Float,Tarrow(Float,Bool)))
            | Greater -> (nullSubst, IGreater, Tarrow(Int,Tarrow(Int,Bool)))
            | GreaterF -> (nullSubst, IGreaterF,
                Tarrow(Float,Tarrow(Float,Bool)))
            | And -> (nullSubst, IAnd, Tarrow(Bool,Tarrow(Bool,Bool)))
            | Or -> (nullSubst, IOr, Tarrow(Bool,Tarrow(Bool,Bool)))
            | Not -> (nullSubst, INot, Tarrow(Bool,Bool))
            | Int_to_float -> (nullSubst, IInt_to_float, Tarrow(Int,Float))
            | Cons -> let polyty = newTyVar "a" in
                (nullSubst, ICons, Tarrow(polyty,
                Tarrow (TconList polyty, TconList polyty)))
            | Cat -> let polyty = newTyVar "a" in
                (nullSubst, ICat, Tarrow(TconList polyty,
                    Tarrow (TconList polyty, TconList polyty)))
            | Len -> let polyty = newTyVar "a" in
                (nullSubst, ILen, Tarrow(TconList polyty, Int))
            | Head -> let polyty = newTyVar "a" in
                (nullSubst, IHead,
                (Tarrow(TconList polyty, polyty)))
            | Tail -> let polyty = newTyVar "a" in
                (nullSubst, ITail,
                Tarrow(TconList polyty, TconList polyty))
            | First -> let polyty1 = newTyVar "a" in
                       let polyty2 = newTyVar "b" in
                       (nullSubst, IFirst,
                       (Tarrow(TconTuple(polyty1,polyty2),polyty1)))
            | Sec -> let polyty1 = newTyVar "a" in
                       let polyty2 = newTyVar "b" in
                       (nullSubst, ISec,
                       (Tarrow(TconTuple(polyty1,polyty2),polyty2)))
                   | Is_none -> let polyty = newTyVar "a" in
                                        (nullSubst, IIs_none,
                                        (Tarrow(Tmaybe polyty, Bool)))
                   | From_just -> let polyty = newTyVar "a" in
                                         (nullSubst, IFrom_just,
                                         (Tarrow(Tmaybe polyty, polyty)))
            | _ -> raise (Failure "not yet implemented in type inference")

let rec typeUpdateEnv env = function
```

```
    | ((a,Vdef(name,expr))::xs) ->
        let (substs, ix, ty) = ti env expr in
        let newTy = generalize env ty in
        let oldTy =
                (match TyEnvMap.find_opt name env with
                | None -> raise(Failure("unbound variable " ^ name))
                | Some si -> instantiate si) in
        let newSubst = mgu newTy oldTy in
        let newPair = (a, InferredVdef(name,
            (composeSubst newSubst substs, ix, apply newSubst ty))) in
            (newPair::(typeUpdateEnv (applyenv newSubst env) xs))
    | [] -> []
    | ((_,Annot(_))::xs) -> raise (Failure "cannot tiVdef on annotation")

let rec unzip_thruple l =
    let f (l1,l2,l3) (x,y,z) = (x::l1,y::l2,z::l3) in
    List.fold_left f ([],[],[]) (List.rev l)

let type_paired_program annotvdef_list =
    let vdef_names = List.fold_left
        (fun l -> fun pair -> (
        match pair with ((Annot(n,_)),_) -> n::l
        | _ -> raise (Failure "vdef where annot should be in pair")) )
        [] annotvdef_list in
    let moduleEnv = List.fold_left
        (fun env -> fun name ->
            let var = newTyVar name in
            TyEnvMap.add name var env)
        TyEnvMap.empty vdef_names in

    let annotIVdefs = typeUpdateEnv moduleEnv annotvdef_list in

    let substList = List.fold_left
        (fun l -> fun (_, InferredVdef(_,(subst,_,_))) -> subst::l)
        [] annotIVdefs in
    let allSubsts = List.fold_left
        (fun s1 -> fun s2 -> composeSubst s1 s2)
        (List.hd substList) substList in
    let annotIVdefs' = List.map
        (fun x -> match x with
                    (Annot(na,tya), InferredVdef(n,(s,ix,ty))) ->
                let finalUnion = mgu tya ty in
                let fullUnion = composeSubst finalUnion allSubsts in
                (Annot(na,tya),
                                InferredVdef(n,(s,ix, apply fullUnion ty)))
                    |_ -> raise (Failure("no"));
                ) annotIVdefs in
        (allSubsts,annotIVdefs')
```

8.12 thunk.ml (Amanda, Hans, Hollis, Da)

```
module L = Llvm
```

```
open Ast
open Tast
open Structs
open Lib

let initThunk_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
        [| L.pointer_type struct_thunk_type
            ; L.pointer_type eval_func_type
                        ; i32_t ; i32_t |]

let initThunk : L.llvalue =
    L.declare_function "init_thunk" initThunk_t the_module

let initThunkLiteral_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
        [| L.pointer_type i8_t |]
let initThunkLiteral : L.llvalue =
    L.declare_function "init_thunk_literal" initThunkLiteral_t the_module

let apply : L.llvalue =
    L.declare_function "apply" call_func_type the_module

let invoke : L.llvalue =
    L.declare_function "invoke" eval_func_type the_module


let struct_thunk = L.declare_global struct_thunk_type "Thunk" the_module
```

8.13 thunk.h (Amanda, Hans, Hollis, Da)

```
#ifndef THUNK
#define THUNK

struct Thunk {
    void *(* eval)(struct Thunk *);
    int num_args;
    int filled_args;
    struct Thunk **args;
    void *value;
    int is_ite;
};

struct Thunk *init_thunk_literal(void *data);
struct Thunk *init_thunk(struct Thunk *t,
    void *(*eval)(struct Thunk *), int num_args, int is_ite);
struct Thunk *apply(struct Thunk *t, struct Thunk *arg);
void *invoke(struct Thunk *t);
#endif
```

8.14 thunk.c (Amanda, Hans, Hollis, Da)

```
#include <string.h>
#include "thunk.h"
#include "lib.h"
#include "natives.h"

struct Thunk *init_thunk(struct Thunk *thunk,
    void *(*eval)(struct Thunk  *), int num_args, int is_ite) {

    thunk->eval = eval;
    thunk->num_args = num_args;
    thunk->filled_args = 0;
    thunk->args = malloc(num_args * sizeof(struct Thunk*));
    thunk->value = NULL;
    thunk ->is_ite = is_ite;

    return thunk;
}

struct Thunk *init_thunk_literal(void *data) {
    struct Thunk *lit = malloc(sizeof(struct Thunk));
    lit = init_thunk(lit,NULL, 1, 0);

    lit->is_ite = 0;
    lit->filled_args = 1;
    lit->value = data;
    (lit->args)[0] = lit;
    return lit;
}

struct Thunk *apply(struct Thunk *in_thunk, struct Thunk *arg) {
        if (in_thunk->eval == NULL) {
                fprintf(stderr, "eval is null\n");
                return apply(in_thunk->value, arg);
        }

        struct Thunk *thunk = invoke(in_thunk);

    struct Thunk *new_thunk = malloc(sizeof(struct Thunk));
    memcpy(new_thunk, thunk, sizeof(struct Thunk));
    new_thunk->args = malloc(new_thunk->num_args * sizeof(struct Thunk*));
    memcpy(new_thunk->args, thunk->args,
        new_thunk->num_args * sizeof(struct Thunk *));

    if (thunk->value == thunk) {
        new_thunk->value = new_thunk;
    }


    if (new_thunk->filled_args < new_thunk->num_args) {
        (new_thunk->args)[new_thunk->filled_args] = arg;
        new_thunk->filled_args++;
```

```
    } else {
                fprintf(stderr, "lets recurse!\n");
                struct Thunk *last_arg_thunk = apply(
                    (new_thunk->args)[new_thunk->num_args - 1], arg);
                new_thunk->args[new_thunk->num_args - 1] = last_arg_thunk;


           /*// Top thunk is filled, let's fill in the last arg
        struct Thunk *last_arg = new_thunk->args[new_thunk->num_args - 1];
        if (last_arg->filled_args < last_arg->num_args) {
            (last_arg->args)[last_arg->filled_args] = arg;
            last_arg->filled_args++;
        } else {
            fprintf(stderr, "fully applied");
            exit(1);
        }*/
    }

        return new_thunk;
}


void *invoke(struct Thunk *t) {
    if (t->filled_args != t->num_args) {
        return (t->value = t);
    } else {
        if (t->value && (t->value != t)) {
            return t->value;
        } else {
            int i;
            for (i = 0; i < t->num_args; i++) {
                    if (!(t->is_ite)) {
                            invoke(t->args[i]);
                }
            }

            t->value = t->eval(t);
            return t->value;
        }
    }
}
```

8.15 mymap.ml (Amanda, Hans)

```
module L = Llvm
open Ast
open Tast
open Structs
open Lib

let mapl_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
```

```
        [| L.pointer_type struct_thunk_type ;
            L.pointer_type struct_thunk_type; i32_t |]
let mapl : L.llvalue =
    L.declare_function "mapl" mapl_t the_module

let filterl : L.llvalue =
    L.declare_function "filterl" mapl_t the_module

let map_listl_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
        [| L.pointer_type struct_thunk_type ;
            L.pointer_type struct_thunk_type ; i32_t |]
let map_listl : L.llvalue =
    L.declare_function "map_listl" map_listl_t the_module
```

8.16 mymap.h (Amanda, Hans)

```
#ifndef MAP
#define MAP


struct Thunk *mapl(struct Thunk *list, struct Thunk *func, int ty);

struct Thunk *filterl(struct Thunk *list, struct Thunk *filter,int ty);

struct Thunk *map_listl(struct Thunk *apps, struct Thunk *vals, int ty);
#endif
```

8.17 mymap.c (Amanda, Da, Hans)

```
mymap.c (Amanda, Da, Hans)
#include <string.h>
#include "lib.h"
#include "mymap.h"
#include "thunk.h"
#include "natives.h"

struct Thunk *mapl(struct Thunk *list_thunk, struct Thunk *func, int ty) {
    struct List *list = invoke(list_thunk);


    struct Thunk *new_thunk = makeEmptyList(ty);
    struct List *new = invoke(new_thunk);

    struct Node *curr = list->head;

    while (curr) {
        struct Thunk *data = (curr->data);
        invoke(func);


        struct Thunk *newThunk = apply(func->value, data);
```

```
        struct Node *newNode = malloc(sizeof(struct Node));
        newNode->data = newThunk;
        newNode->next = NULL;
        appendNode(new, newNode);
        curr = curr->next;
    }
    return init_thunk_literal(new);
}

struct Thunk *map_listl(struct Thunk *apps_thunk, struct Thunk *vals,
    int ty) {
    struct List *apps = invoke(apps_thunk);

        struct Thunk *new_thunk = makeEmptyList(ty);
        struct List *new = invoke(new_thunk);

    struct Node *curr_app_node = apps->head;

    while (curr_app_node) {
        struct Thunk *curr_app = curr_app_node->data;
        struct Thunk *applied_thunk = mapl(vals, curr_app, ty);

        struct Thunk *stupid_thunk_list_wrapper_new
            = init_thunk_literal(new);

        new = cat(stupid_thunk_list_wrapper_new,applied_thunk);
        curr_app_node = curr_app_node->next;
    }


    return init_thunk_literal(new);
}

struct Thunk *filterl(struct Thunk *list_thunk, struct Thunk *filter,
    int ty) {
    struct List *list = invoke(list_thunk);

    struct Thunk *new_thunk = makeEmptyList(ty);
    struct List *new = invoke(new_thunk);

    struct Node *curr = list->head;

    while (curr) {
        // thunk inside list
        struct Thunk *currThunk = curr->data;
        // number of args stored inside list
        int list_filled_args = currThunk->filled_args;
        int list_num_args = currThunk->num_args;
        int filter_num_args = filter->num_args;
        int filter_filled_args = filter->filled_args;
```

```
            int start_offset = list_num_args -
                (filter_num_args-filter_filled_args);

            struct Thunk **args = currThunk->args;
            struct Thunk **start = args + start_offset;

            struct Thunk *passesFilter = filter;

            int count = list_filled_args - (start -args);

            while (count) {
                passesFilter = apply(passesFilter, *start);
                start++;
                count--;
                /*
                passesFilter = apply(passesFilter, *args);
                args++;
                filled_args--;
                */
            }

            void *value = invoke(passesFilter);


            int passed = *(int *)value;

            if (passed) {
                struct Node *newNode = malloc(sizeof(struct Node));
                newNode->data = curr->data;
                newNode->next = NULL;
                appendNode(new,newNode);
            }
            curr = curr->next;
        }
        return init_thunk_literal(new);
    }
```

8.18 natives.h (Amanda, Hans)

```
natives.h (Amanda, Hans)
#include "lib.h"
#include "mymap.h"
#include "thunk.h"


// Integer operations
int *add(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *add_eval(struct Thunk *t);
extern struct Thunk add_init_thunk[1];

int *sub(struct Thunk *x_thunk, struct Thunk *y_thunk);
```

```
void *sub_eval(struct Thunk *t);
extern struct Thunk sub_init_thunk[1];

int *mult(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *mult_eval(struct Thunk *t);
extern struct Thunk mult_init_thunk[1];

int *divi(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *divi_eval(struct Thunk *t);
extern struct Thunk divi_init_thunk[1];

int *mod(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *mod_eval(struct Thunk *t);
extern struct Thunk mod_init_thunk[1];

int *powe(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *powe_eval(struct Thunk *t);
extern struct Thunk powe_init_thunk[1];

int *eq(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *eq_eval(struct Thunk *t);
extern struct Thunk eq_init_thunk[1];

int *neq(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *neq_eval(struct Thunk *t);
extern struct Thunk neq_init_thunk[1];

int *geq(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *geq_eval(struct Thunk *t);
extern struct Thunk geq_init_thunk[1];

int *leq(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *leq_eval(struct Thunk *t);
extern struct Thunk leq_init_thunk[1];

int *less(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *less_eval(struct Thunk *t);
extern struct Thunk less_init_thunk[1];

int *greater(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *greater_eval(struct Thunk *t);
extern struct Thunk greater_init_thunk[1];

int *neg(struct Thunk *x_thunk);
void *neg_eval(struct Thunk *t);
extern struct Thunk neg_init_thunk[1];


// Float operations
float *addf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *addf_eval(struct Thunk *t);
```

```
extern struct Thunk addf_init_thunk[1];

float *subf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *subf_eval(struct Thunk *t);
extern struct Thunk subf_init_thunk[1];

float *multf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *multf_eval(struct Thunk *t);
extern struct Thunk multf_init_thunk[1];

float *divf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *divf_eval(struct Thunk *t);
extern struct Thunk divf_init_thunk[1];

float *powef(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *powef_eval(struct Thunk *t);
extern struct Thunk powf_init_thunk[1];

int *eqf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *eqf_eval(struct Thunk *t);
extern struct Thunk eqf_init_thunk[1];

int *neqf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *neqf_eval(struct Thunk *t);
extern struct Thunk neqf_init_thunk[1];

int *geqf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *geqf_eval(struct Thunk *t);
extern struct Thunk geqf_init_thunk[1];

int *leqf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *leqf_eval(struct Thunk *t);
extern struct Thunk leqf_init_thunk[1];

int *lessf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *lessf_eval(struct Thunk *t);
extern struct Thunk lessf_init_thunk[1];

int *greaterf(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *greaterf_eval(struct Thunk *t);
extern struct Thunk greaterf_init_thunk[1];

float *negf(struct Thunk *x_thunk);
void *negf_eval(struct Thunk *t);
extern struct Thunk negf_init_thunk[1];

// Boolean operations
int *andb(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *andb_eval(struct Thunk *t);
extern struct Thunk andb_init_thunk[1];
```

```
int *orb(struct Thunk *x_thunk, struct Thunk *y_thunk);
void *orb_eval(struct Thunk *t);
extern struct Thunk orb_init_thunk[1];

int *notb(struct Thunk *x_thunk);
void *notb_eval(struct Thunk *t);
extern struct Thunk notb_init_thunk[1];


// List operations
struct List *cons(struct Thunk *data_thunk, struct Thunk *list_thunk);
void *cons_eval(struct Thunk *);
extern struct Thunk cons_init_thunk[1];

struct List *cat(struct Thunk *lthunk1, struct Thunk *lthunk2);
void *cat_eval(struct Thunk *t);
extern struct Thunk cat_init_thunk[1];

int *length(struct Thunk *lthunk);
void *length_eval(struct Thunk *t);
extern struct Thunk length_init_thunk[1];

void *head(struct Thunk *lthunk);
void *head_eval(struct Thunk *t);
extern struct Thunk head_init_thunk[1];

struct List *tail(struct Thunk *lthunk);
void *tail_eval(struct Thunk *lthunk);
extern struct Thunk tail_init_thunk[1];

// Tuple operations
void *first(struct Thunk *lthunk);
void *first_eval(struct Thunk *t);
extern struct Thunk first_init_thunk[1];

void *second(struct Thunk *lthunk);
void *second_eval(struct Thunk *t);
extern struct Thunk second_init_thunk[1];

// maybe operations
void *isNone(struct Thunk *the_thunk);
void *is_none_eval(struct Thunk *t);
extern struct Thunk is_none_init_thunk[1];

void *fromJust(struct Thunk *th_thunk);
void *from_just_eval(struct Thunk *t);
extern struct Thunk from_just_init_thunk[1];

void *int_to_float(struct Thunk *t);
void *int_to_float_eval(struct Thunk *t);
extern struct Thunk int_to_float_init_thunk[1];
```

```
void *ite(struct Thunk *c, struct Thunk *t, struct Thunk *e);
void *ite_eval(struct Thunk *t);
extern struct Thunk ite_init_thunk[1];
```

8.19 natives.c (Hans, Amanda, Hollis, Da)

```
#include <string.h>
#include "lib.h"
#include "mymap.h"
#include "thunk.h"
#include "natives.h"
#include <math.h>

// Integer operations
int *add(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *x = invoke(x_thunk);
    void *y = invoke(y_thunk);


    int x_ = *(int *)x;
    int y_ = *(int *)y;
    int local = x_ + y_;

    int *result = malloc(sizeof(int));


    *result = local;
    return result;
}

void *add_eval(struct Thunk *t) {
    struct Thunk *x_ = ((t->args)[0]);
    struct Thunk *y_ = ((t->args)[1]);

    int * result = add(x_,y_);


    return result;
}

int *sub(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *x = invoke(x_thunk);
    void *y = invoke(y_thunk);


    int x_ = *(int *)x;
    int y_ = *(int *)y;
    int local = x_ - y_;

    int *result = malloc(sizeof(int));
```

```
    *result = local;
    return result;
}

void *sub_eval(struct Thunk *t) {
    struct Thunk *x_ = ((t->args)[0]);
    struct Thunk *y_ = ((t->args)[1]);

    int * result = sub(x_,y_);


    return result;
}

int *mult(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);


    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ * y_;

    return result;
}

void *mult_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = mult(x,y);

    return result;
}

int *divi(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);


    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ / y_;

    return result;
}
```

```
void *divi_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = divi(x,y);

    return result;
}

int *mod(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);


    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ % y_;

    return result;
}

void *mod_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = mod(x,y);

    return result;
}

int *powe(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);


    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = (int) (pow((double) x_, (double) y_));

    return result;
}

void *powe_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);
```

```
    void *result = powe(x,y);

    return result;
}

int *eq(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ == y_;

    return result;
}

void *eq_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = eq(x,y);


    return result;
}

int *neq(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ != y_;

    return result;
}

void *neq_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = neq(x,y);


    return result;
}
```

```
int *geq(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ >= y_;

    return result;
}

void *geq_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = geq(x,y);


    return result;
}

int *leq(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ <= y_;

    return result;
}

void *leq_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = leq(x,y);


    return result;
}

int *less(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    int x_ = *(int *)data1;
```

```
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ < y_;

    return result;
}

void *less_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = less(x,y);


    return result;
}

int *greater(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ > y_;

    return result;
}

void *greater_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = greater(x,y);


    return result;
}

int *neg(struct Thunk *x_thunk) {
    void *data1 = invoke(x_thunk);

    int x_ = *(int *)data1;

    int *result = malloc(sizeof(int));
    *result = -x_;

    return result;
}
```

```
void *neg_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);

    void *result = neg(x);


    return result;
}


// Float operations
float *addf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *x = invoke(x_thunk);
    void *y = invoke(y_thunk);


    float x_ = *(float *)x;
    float y_ = *(float *)y;
    float local = x_ + y_;

    float *result = malloc(sizeof(float));


    *result = local;
    return result;
}

void *addf_eval(struct Thunk *t) {
    struct Thunk *x_ = ((t->args)[0]);
    struct Thunk *y_ = ((t->args)[1]);

    void *result = addf(x_,y_);


    return result;
}

float *subf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *x = invoke(x_thunk);
    void *y = invoke(y_thunk);


    float x_ = *(float *)x;
    float y_ = *(float *)y;
    float local = x_ - y_;

    float *result = malloc(sizeof(float));


    *result = local;
```

```
    return result;
}

void *subf_eval(struct Thunk *t) {
    struct Thunk *x_ = ((t->args)[0]);
    struct Thunk *y_ = ((t->args)[1]);

    void *result = subf(x_,y_);


    return result;
}

float *multf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    float x_ = *(float *)data1;
    float y_ = *(float *)data2;

    float *result = malloc(sizeof(float));
    *result = x_ * y_;

    return result;
}

void *multf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = multf(x,y);

    return result;
}

float *divf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);


    float x_ = *(float *)data1;
    float y_ = *(float *)data2;

    float *result = malloc(sizeof(float));
    *result = x_ / y_;

    return result;
}

void *divf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
```

```
    struct Thunk *y = ((t->args)[1]);

    void *result = divf(x,y);

    return result;
}

float *powef(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);


    float x_ = *(float *)data1;
    float y_ = *(float *)data2;

    float *result = malloc(sizeof(float));

    *result = (float) (pow((double) x_, (double) y_));

    return result;
}

void *powef_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = powef(x,y);

    return result;
}

int *eqf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    float x_ = *(float *)data1;
    float y_ = *(float *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ == y_;

    return result;
}

void *eqf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = eqf(x,y);
```

```
    return result;
}

int *neqf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    float x_ = *(float *)data1;
    float y_ = *(float *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ != y_;

    return result;
}

void *neqf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = neqf(x,y);


    return result;
}

int *geqf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    float x_ = *(float *)data1;
    float y_ = *(float *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ >= y_;

    return result;
}

void *geqf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = geqf(x,y);


    return result;
}

int *leqf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
```

```
    void *data2 = invoke(y_thunk);

    float x_ = *(float *)data1;
    float y_ = *(float *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ <= y_;

    return result;
}

void *leqf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = leqf(x,y);


    return result;
}

int *lessf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    float x_ = *(float *)data1;
    float y_ = *(float *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ < y_;

    return result;
}

void *lessf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = lessf(x,y);


    return result;
}

int *greaterf(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    float x_ = *(float *)data1;
    float y_ = *(float *)data2;
```

```
    int *result = malloc(sizeof(int));
    *result = x_ > y_;

    return result;
}

void *greaterf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = greaterf(x,y);


    return result;
}

float *negf(struct Thunk *x_thunk) {
    void *data1 = invoke(x_thunk);

    float x_ = *(float *)data1;

    float *result = malloc(sizeof(float));
    *result = -x_;

    return result;
}

void *negf_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);

    void *result = negf(x);


    return result;
}

// Boolean operations
int *andb(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = invoke(x_thunk);
    void *data2 = invoke(y_thunk);

    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ && y_;

    return result;
}

void *andb_eval(struct Thunk *t) {
```

```
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = andb(x,y);


    return result;
}

int *orb(struct Thunk *x_thunk, struct Thunk *y_thunk) {
    void *data1 = x_thunk->value;
    void *data2 = y_thunk->value;

    int x_ = *(int *)data1;
    int y_ = *(int *)data2;

    int *result = malloc(sizeof(int));
    *result = x_ || y_;

    return result;
}

void *orb_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);
    struct Thunk *y = ((t->args)[1]);

    void *result = orb(x,y);


    return result;
}

int *notb(struct Thunk *x_thunk) {
    void *data1 = invoke(x_thunk);

    int x_ = *(int *)data1;

    int *result = malloc(sizeof(int));
    *result = !x_;

    return result;
}

void *notb_eval(struct Thunk *t) {
    struct Thunk *x = ((t->args)[0]);

    void *result = notb(x);


    return result;
}
```

```
// List operations
struct List *cons(struct Thunk *data_thunk, struct Thunk *list_thunk) {
    struct List *list = list_thunk->value;

    struct Node *newhead = malloc(sizeof(struct Node));
    newhead->data = data_thunk;


    struct List *newlist = malloc(sizeof(struct List));
    memcpy(newlist, list, sizeof(struct List));


    newlist->head = newhead;
    newlist->last_eval = newhead;

    struct Node *curr = list->head;

    while(curr) {
        struct Node *newnode = malloc(sizeof(struct Node));
        newnode->next = NULL;
        newnode->data = curr->data;
        appendNode(newlist, newnode);
        curr = curr->next;
    }

    return newlist;
}

void *cons_eval(struct Thunk *t) {
    struct Thunk *data_thunk = ((t->args)[0]);
    struct Thunk *list_thunk = ((t->args)[1]);

    struct List *result = cons(data_thunk, list_thunk);
    return result;
}

struct List *cat(struct Thunk *lthunk1, struct Thunk *lthunk2) {
    struct List *l1 = invoke(lthunk1);
    struct List *l2 = invoke(lthunk2);

    struct List *new = malloc(sizeof(struct List));
    memcpy(new,l2,sizeof(struct List));
    new->head = NULL;
    new->last_eval = NULL;

    struct Node *curr1 = l1->head;
    while (curr1) {
        struct Node *newnode = malloc(sizeof(struct Node));
        newnode->data = curr1->data;
```

```
        newnode->next = NULL;
        appendNode(new, newnode);
        curr1 = curr1->next;
    }

    struct Node *curr2 = l2->head;
    while (curr2) {
        struct Node *newnode = malloc(sizeof(struct Node));
        newnode->data = curr2->data;
        newnode->next = NULL;
        appendNode(new, newnode);
        curr2 = curr2->next;
    }

    return new;
}

void *cat_eval(struct Thunk *t) {
    struct Thunk *lthunk1 = ((t->args)[0]);
    struct Thunk *lthunk2 = ((t->args)[1]);

    struct List *result = cat(lthunk1, lthunk2);

    return result;
}

int *length(struct Thunk *lthunk) {
    struct List *list = invoke(lthunk);

    struct Node *curr = list->head;
    int count = 0;
    while (curr) {
        count++;
        curr = curr->next;
    }
    int *result = malloc(sizeof(int));
    *result = count;
    return result;
}

void *length_eval(struct Thunk *t) {
    struct Thunk *lthunk = ((t->args)[0]);
    int *result = length(lthunk);
    return result;
}

void *head(struct Thunk *lthunk) {
    struct List *list = invoke(lthunk);


    struct Thunk *data = (list->head)->data;
```

```
    void *value = invoke(data);
    return value;
}

void *head_eval(struct Thunk *t) {
    struct Thunk *lthunk = ((t->args)[0]);
    void *result = head(lthunk);
    return result;
}

struct List *tail(struct Thunk *lthunk) {
    struct List *list = invoke(lthunk);
    struct List *newlist = malloc(sizeof(struct List));
    memcpy(newlist, list, sizeof(struct List));
    newlist->head = NULL;
    newlist->last_eval = NULL;


    struct Node *curr = list->head;
    if (!curr)
        return newlist;

    curr = curr->next;
    while (curr) {
        struct Thunk *data = curr->data;

        struct Node *newnode = malloc(sizeof(struct Node));
        newnode->next = NULL;
        newnode->data = curr->data;

        appendNode(newlist,newnode);

        curr = curr->next;
    }
    return newlist;
}

void *tail_eval(struct Thunk *t) {
    struct Thunk *lthunk = ((t->args)[0]);
    struct List *result = tail(lthunk);
    return result;
}


// Tuple operations
void *first(struct Thunk *lthunk) {
    struct Tuple *tuple = invoke(lthunk);
    struct Thunk *data = tuple->first;
    void *value = invoke(data);
    return value;
}
```

```
void *first_eval(struct Thunk *t) {
    struct Thunk *lthunk = ((t->args)[0]);
    void *result = first(lthunk);
    return result;
}

void *second(struct Thunk *lthunk) {
    struct Tuple *tuple = invoke(lthunk);
    struct Thunk *data = tuple->second;
    void *value = invoke(data);
    return value;
}

void *second_eval(struct Thunk *t) {
    struct Thunk *lthunk = ((t->args)[0]);
    void *result = second(lthunk);
    return result;
}

void *isNone(struct Thunk *the_thunk){
        struct Maybe *mb = invoke(the_thunk);
    int is_none = mb -> is_none;
    char *is_none_char = malloc(sizeof(char));
    *is_none_char = (char)is_none;
    return is_none_char;
}

void *fromJust(struct Thunk *the_thunk){
    struct Maybe *mb = invoke(the_thunk);
    struct Thunk *data = mb -> data;
    void *value = invoke(data);
    return value;
}

void *from_just_eval(struct Thunk *t){
    struct Thunk *argy = ((t -> args)[0]);
    void *result = fromJust(argy);
    return result;
}


void *is_none_eval(struct Thunk *t){
    struct Thunk *argy = ((t->args)[0]);
    void *result = isNone(argy);
    return result;
}


void *int_to_float(struct Thunk *i) {
    int *val = invoke(i);
```

```
    float *result = malloc(sizeof(float));
    *result = *(int *)val;
    return result;
}

void *int_to_float_eval(struct Thunk *t) {
    struct Thunk *th = ((t->args)[0]);
    void *result = int_to_float(th);
    return result;
}

void *ite(struct Thunk *cond_thunk, struct Thunk *then_thunk,
    struct Thunk *else_thunk){
    void *val = invoke(cond_thunk);
    char boolean_val = *(char *)(val);
    if (boolean_val){
        return invoke(then_thunk);
    }
    else {
        return invoke(else_thunk);
    }
}

void *ite_eval(struct Thunk *t){
    struct Thunk *cond_thunk = ((t->args)[0]);
    struct Thunk *then_thunk = ((t->args)[1]);
    struct Thunk *else_thunk = ((t->args)[2]);
    //struct Thunk* result = malloc(sizeof(struct Thunk));
    return ite(cond_thunk, then_thunk, else_thunk);
}
```

8.20 lib.ml (Amanda, Hans, Hollis, Da)

```
module L = Llvm
open Ast
open Tast
open Structs

let ty_code = function
    | Int           -> 0
    | Bool          -> 1
    | Char          -> 2
    | Float         -> 3
    | TconList _            -> 4
    | TconTuple _           -> 5
    | Tmaybe _          -> 6
    | _         -> raise (Failure "no type code: unprintable")


let the_module = L.create_module context "Rippl"

let main_t = L.function_type i32_t [| |]
```

```
let main_f = L.define_function "main" main_t the_module

let builder = L.builder_at_end context (L.entry_block main_f)

let char_format_str = L.build_global_stringptr "%c" "fmt" builder
let int_format_str = L.build_global_stringptr "%d" "fmt_int" builder
let float_format_str = L.build_global_stringptr "%f" "fmt_float" builder

(* Integer operation thunks *)
let add_init_thunk = L.define_global "add_init_thunk"
    (L.const_null struct_thunk_type) the_module
let sub_init_thunk = L.define_global "sub_init_thunk"
    (L.const_null struct_thunk_type) the_module
let mult_init_thunk = L.define_global "mult_init_thunk"
    (L.const_null struct_thunk_type) the_module
let divi_init_thunk = L.define_global "divi_init_thunk"
    (L.const_null struct_thunk_type) the_module
let mod_init_thunk = L.define_global "mod_init_thunk"
    (L.const_null struct_thunk_type) the_module
let powe_init_thunk = L.define_global "powe_init_thunk"
    (L.const_null struct_thunk_type) the_module
let eq_init_thunk = L.define_global "eq_init_thunk"
    (L.const_null struct_thunk_type) the_module
let neq_init_thunk = L.define_global "neq_init_thunk"
    (L.const_null struct_thunk_type) the_module
let geq_init_thunk = L.define_global "geq_init_thunk"
    (L.const_null struct_thunk_type) the_module
let leq_init_thunk = L.define_global "leq_init_thunk"
    (L.const_null struct_thunk_type) the_module
let less_init_thunk = L.define_global "less_init_thunk"
    (L.const_null struct_thunk_type) the_module
let greater_init_thunk = L.define_global "greater_init_thunk"
    (L.const_null struct_thunk_type) the_module
let neg_init_thunk = L.define_global "neg_init_thunk"
    (L.const_null struct_thunk_type) the_module

(* Float operation thunks *)
let addf_init_thunk = L.define_global "addf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let subf_init_thunk = L.define_global "subf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let multf_init_thunk = L.define_global "multf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let divf_init_thunk = L.define_global "divf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let powef_init_thunk = L.define_global "powef_init_thunk"
    (L.const_null struct_thunk_type) the_module
let eqf_init_thunk = L.define_global "eqf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let neqf_init_thunk = L.define_global "neqf_init_thunk"
    (L.const_null struct_thunk_type) the_module
```

```
let geqf_init_thunk = L.define_global "geqf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let leqf_init_thunk = L.define_global "leqf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let lessf_init_thunk = L.define_global "lessf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let greaterf_init_thunk = L.define_global "greaterf_init_thunk"
    (L.const_null struct_thunk_type) the_module
let negf_init_thunk = L.define_global "negf_init_thunk"
    (L.const_null struct_thunk_type) the_module

(* Boolean operation thunks *)
let andb_init_thunk = L.define_global "andb_init_thunk"
    (L.const_null struct_thunk_type) the_module
let orb_init_thunk = L.define_global "orb_init_thunk"
    (L.const_null struct_thunk_type) the_module
let notb_init_thunk = L.define_global "notb_init_thunk"
    (L.const_null struct_thunk_type) the_module

(* List operations *)
let cons_init_thunk = L.define_global "cons_init_thunk"
    (L.const_null struct_thunk_type) the_module
let cat_init_thunk = L.define_global "cat_init_thunk"
    (L.const_null struct_thunk_type) the_module
let length_init_thunk = L.define_global "length_init_thunk"
    (L.const_null struct_thunk_type) the_module
let head_init_thunk = L.define_global "head_init_thunk"
    (L.const_null struct_thunk_type) the_module
let tail_init_thunk = L.define_global "tail_init_thunk"
    (L.const_null struct_thunk_type) the_module

(* Maybe operations *)
let is_none_init_thunk = L.define_global "is_none_init_thunk"
    (L.const_null struct_thunk_type) the_module
let from_just_init_thunk = L.define_global "from_just_init_thunk"
    (L.const_null struct_thunk_type) the_module

(* Tuple operations *)
let first_init_thunk = L.define_global "first_init_thunk" (L.const_null
    struct_thunk_type) the_module
let second_init_thunk = L.define_global "second_init_thunk" (L.const_null
    struct_thunk_type) the_module

let ite_init_thunk = L.define_global "ite_init_thunk" (L.const_null
    struct_thunk_type) the_module

let int_to_float_init_thunk = L.define_global "int_to_float_init_thunk"
    (L.const_null
    struct_thunk_type) the_module

let _ = L.set_alignment 32 add_init_thunk;
```

```
        L.set_alignment 32 sub_init_thunk;
        L.set_alignment 32 mult_init_thunk;
        L.set_alignment 32 divi_init_thunk;
        L.set_alignment 32 mod_init_thunk;
        L.set_alignment 32 powe_init_thunk;
        L.set_alignment 32 eq_init_thunk;
        L.set_alignment 32 neq_init_thunk;
        L.set_alignment 32 geq_init_thunk;
        L.set_alignment 32 leq_init_thunk;
        L.set_alignment 32 less_init_thunk;
        L.set_alignment 32 greater_init_thunk;
        L.set_alignment 32 neg_init_thunk;

        L.set_alignment 32 addf_init_thunk;
        L.set_alignment 32 subf_init_thunk;
        L.set_alignment 32 multf_init_thunk;
        L.set_alignment 32 divf_init_thunk;
        L.set_alignment 32 powef_init_thunk;
        L.set_alignment 32 eqf_init_thunk;
        L.set_alignment 32 neqf_init_thunk;
        L.set_alignment 32 geqf_init_thunk;
        L.set_alignment 32 leqf_init_thunk;
        L.set_alignment 32 lessf_init_thunk;
        L.set_alignment 32 greaterf_init_thunk;
        L.set_alignment 32 negf_init_thunk;

        L.set_alignment 32 andb_init_thunk;
        L.set_alignment 32 orb_init_thunk;
        L.set_alignment 32 notb_init_thunk;

        L.set_alignment 32 cons_init_thunk;
        L.set_alignment 32 cat_init_thunk;
        L.set_alignment 32 length_init_thunk;
        L.set_alignment 32 head_init_thunk;
        L.set_alignment 32 tail_init_thunk;


        L.set_alignment 32 first_init_thunk;
        L.set_alignment 32 second_init_thunk;

        L.set_alignment 32 is_none_init_thunk;
        L.set_alignment 32 from_just_init_thunk;

        L.set_alignment 32 int_to_float_init_thunk;
        L.set_alignment 32 ite_init_thunk

let l_char = L.const_int i8_t (Char.code '\n')

(* HEAP ALLOCATE PRIMS *)
let makeInt_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type) [| i32_t |]
```

```
let makeInt : L.llvalue =
    L.declare_function "makeInt" makeInt_t the_module
let makeBool_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type) [| i8_t |]
let makeBool : L.llvalue =
    L.declare_function "makeBool" makeBool_t the_module
let makeChar_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type) [| i8_t |]
let makeChar : L.llvalue =
    L.declare_function "makeChar" makeChar_t the_module
let makeFloat_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type) [| float_t |]
let makeFloat : L.llvalue =
    L.declare_function "makeFloat" makeFloat_t the_module

(* HEAP ALLOCATE TUPLES *)
let makeTuple_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
    [| L.pointer_type struct_thunk_type ; L.pointer_type struct_thunk_type
        ; i32_t ; i32_t |]
let makeTuple : L.llvalue =
    L.declare_function "makeTuple" makeTuple_t the_module

(* HEAP ALLOCATE MAYBES *)
let makeMaybe_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
    [| L.pointer_type struct_thunk_type ; i32_t |]
let makeMaybe : L.llvalue =
    L.declare_function "makeMaybe" makeMaybe_t the_module

(* HEAP ALLOCATE LIST STRUCTS *)
let makeNode_t : L.lltype =
    L.function_type (L.pointer_type struct_node_type)
    [| L.pointer_type struct_thunk_type |]
let makeNode : L.llvalue =
    L.declare_function "makeNode" makeNode_t the_module
(* TODO: make node *)

let makeEmptyList_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
    [| i32_t |]
let makeEmptyList : L.llvalue =
    L.declare_function "makeEmptyList" makeEmptyList_t the_module
let makeemptylist ty name =
    L.build_call makeEmptyList
    [| L.const_int i32_t ty |]
    name builder

let makeInfinite_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
    [| i32_t |]
```

```
let makeInfinite : L.llvalue =
    L.declare_function "makeInfinite" makeInfinite_t the_module
let makeinfinite start name = match start with
    | (TIntLit s, _) ->
        L.build_call makeInfinite
        [| L.const_int i32_t s |]
        name builder
    | _ -> raise (Failure "type error in infinite list")

let makeRangeList_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
    [| L.pointer_type struct_thunk_type ; L.pointer_type struct_thunk_type |]
let makeRangeList : L.llvalue =
    L.declare_function "makeRangeList" makeRangeList_t the_module

let makeEmptyList_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
    [|  i32_t |]
let makeEmptyList : L.llvalue =
    L.declare_function "makeEmptyList" makeEmptyList_t the_module

let appendNode_t : L.lltype =
    L.function_type (L.pointer_type struct_list_type)
    [| L.pointer_type struct_list_type ; L.pointer_type struct_node_type |]
let appendNode : L.llvalue =
    L.declare_function "appendNode" appendNode_t the_module
let appendNodeThunk_t : L.lltype =
    L.function_type (L.pointer_type struct_thunk_type)
    [| L.pointer_type struct_thunk_type ; L.pointer_type struct_node_type |]
let appendNodeThunk : L.llvalue =
    L.declare_function "appendNodeThunk" appendNodeThunk_t the_module

(* PRINTING *)
let printf_t : L.lltype =
    L.var_arg_function_type i32_t [| L.pointer_type i8_t |]
let printf_func : L.llvalue =
    L.declare_function "printf" printf_t the_module

let printRangeList_t : L.lltype =
    L.function_type void_t [| L.pointer_type struct_thunk_type |]
let printRangeList : L.llvalue =
    L.declare_function "printRangeList" printRangeList_t the_module

let printPrim_t : L.lltype =
    L.function_type void_t [| L.pointer_type i8_t ; i32_t |]
let printPrim : L.llvalue =
    L.declare_function "printPrim" printPrim_t the_module

let printAnyThunk_t : L.lltype =
    L.function_type void_t [| L.pointer_type struct_thunk_type
    ; L.pointer_type i32_t ; i32_t |]
```

```
let printAnyThunk : L.llvalue =
    L.declare_function "printAnyThunk" printAnyThunk_t the_module

let printBool_t : L.lltype =
    L.function_type void_t [| i8_t |]
let printBool : L.llvalue =
    L.declare_function "printBool" printBool_t the_module
(*let makeIte_t : L.lltype =
        L.function_type (L.pointer_type struct_thunk_type)
        [| L.pointer_type struct_thunk_type ;
        L.pointer_type struct_thunk_type ;
        L.pointer_type struct_thunk_type |]
let makeIte : L.llvalue =
        L.declare_function "makeIte" makeIte_t the_module*)



let initNativeThunks_t : L.lltype =
    L.function_type void_t [| |]
let initNativeThunks : L.llvalue =
    L.declare_function "initNativeThunks" initNativeThunks_t the_module

let _ = L.build_call initNativeThunks [| |] "" builder
```

8.21 lib.h (Amanda, Hans, Hollis, Da)

```
#ifndef LIB
#define LIB

#include <stdio.h>
#include <stdlib.h>

#define INT         0
#define BOOL        1
#define CHAR        2
#define FLOAT       3
#define LIST        4
#define TUPLE       5
#define MAYBE       6

#define RANGE       0
#define INFINITE    1
#define COMP        2
#define LITLIST     3

struct Tuple {
    int t1;
    int t2;
    struct Thunk *first;
    struct Thunk *second;
};
```

```
struct Maybe {
    int ty;
    int is_none;
    struct Thunk *data;
};

struct Node {
    struct Thunk *data;
    struct Node *next;
};

struct List {
    struct Node *head;
    int content_type;
    int type;

    /* INDEXING ACCESS STUFF */
    struct Node *last_eval;
    int curr_index;             // useful for laziness in ranges/infinites
    int start;                  // useful for ranges/infintes
    int end;                    // useful for ranges


};

void printBool(char b);
void printPrim(void *data, int ty);
void printAny(void *thing, int ty);
void printAnyThunk(struct Thunk *t, int *ty, int index);
void printList(struct List *list, int *ty, int index);
void printTuple(void *tup, int *ty, int index);
void printMaybe(void *may, int *ty, int index);
void printRangeList(struct Thunk *list);
void printCompList(void *list);
void printMaybe(void *mayb, int *ty, int index);

struct Thunk *makeInt(int x);
struct Thunk *makeBool(char x);
struct Thunk *makeChar(char x);
struct Thunk *makeFloat(float x);
struct Node *makeNode(struct Thunk *data);
struct Thunk *makeEmptyList(int ty);
struct Thunk *makeInfinite(int start);
struct Thunk *makeRangeList(struct Thunk *start, struct Thunk *end);
struct Thunk *makeTuple(struct Thunk *data1, struct Thunk *data2,
    int t1, int t2);
struct Thunk *makeMaybe(struct Thunk *data, int ty);

void explodeRangeList(void *list);
void evalNextNode(void *list);
struct List *appendNode(struct List *list, struct Node *node);
```

```
struct Thunk *appendNodeThunk(struct Thunk *list, struct Node *node);
struct Node *evalNextNodeComp(void *list, int num_vbinds);
void initNativeThunks();

struct Thunk *makeIte(struct Thunk *cond, struct Thunk *then_thunk,
    struct Thunk *else_thunk);

#endif
```

8.22 lib.c (Amanda, Hans, Hollis, Da)

```
#include <stdio.h>
#include "lib.h"
#include "thunk.h"
#include "mymap.h"
#include <string.h>
#include "natives.h"

struct Thunk add_init_thunk[1];
struct Thunk sub_init_thunk[1];
struct Thunk mult_init_thunk[1];
struct Thunk divi_init_thunk[1];
struct Thunk mod_init_thunk[1];
struct Thunk powe_init_thunk[1];
struct Thunk eq_init_thunk[1];
struct Thunk neq_init_thunk[1];
struct Thunk geq_init_thunk[1];
struct Thunk leq_init_thunk[1];
struct Thunk less_init_thunk[1];
struct Thunk greater_init_thunk[1];
struct Thunk neg_init_thunk[1];

struct Thunk addf_init_thunk[1];
struct Thunk subf_init_thunk[1];
struct Thunk multf_init_thunk[1];
struct Thunk divf_init_thunk[1];
struct Thunk powef_init_thunk[1];
struct Thunk eqf_init_thunk[1];
struct Thunk neqf_init_thunk[1];
struct Thunk geqf_init_thunk[1];
struct Thunk leqf_init_thunk[1];
struct Thunk lessf_init_thunk[1];
struct Thunk greaterf_init_thunk[1];
struct Thunk negf_init_thunk[1];

struct Thunk andb_init_thunk[1];
struct Thunk orb_init_thunk[1];
struct Thunk notb_init_thunk[1];

struct Thunk cons_init_thunk[1];
struct Thunk cat_init_thunk[1];
struct Thunk length_init_thunk[1];
```

```
struct Thunk head_init_thunk[1];
struct Thunk tail_init_thunk[1];

struct Thunk first_init_thunk[1];
struct Thunk second_init_thunk[1];

struct Thunk is_none_init_thunk[1];
struct Thunk from_just_init_thunk[1];

struct Thunk int_to_float_init_thunk[1];

struct Thunk ite_init_thunk[1];

struct Thunk *makeInt(int x) {
    int *i = malloc(sizeof(int));
    *i = x;
    return init_thunk_literal(i);
}

struct Thunk *makeBool(char x) {
    return makeChar(x);
}

struct Thunk *makeChar(char x) {
    char *b = malloc(sizeof(char));
    *b = x;
    return init_thunk_literal(b);
}

struct Thunk *makeFloat(float x) {
    float *f = malloc(sizeof(float));
    *f = x;
    return init_thunk_literal(f);
}

struct Thunk *makeTuple(struct Thunk *data1, struct Thunk *data2,
    int t1, int t2) {
        struct Tuple *newtup = malloc(sizeof(struct Tuple));

    newtup->t1 = t1;
    newtup->t2 = t2;

    newtup->first = data1;
    newtup->second = data2;

    return init_thunk_literal(newtup);
}

struct Thunk *makeMaybe(struct Thunk *data, int ty) {
    struct Maybe *may = malloc(sizeof(struct Maybe));
    if (data) {
```

```
        may->is_none = 0;
    } else {
        may->is_none = 1;
    }

    may->data = data;
    return init_thunk_literal(may);
}


struct Thunk *makeEmptyList(int ty) {
    struct List *new = malloc(sizeof(struct List));
    memset(new,0,sizeof(struct List));

    new->start = 0;
    new->end = 0;
    new->curr_index = -1;
    new->last_eval = NULL;
    new->type = LITLIST;
    new->content_type = ty;



    return init_thunk_literal(new);
}

struct Thunk *makeRangeList(struct Thunk *start, struct Thunk *end) {
    struct List *list = malloc(sizeof(struct List));
    list->start = *(int *)(invoke(start));
    list->end = *(int *)(invoke(end));
    list->content_type = INT;
    list->type = RANGE;
    list->curr_index = list->start;

    struct Thunk *data = makeInt(list->start);
    list->head = makeNode(data);
    list->last_eval = list->head;

    explodeRangeList(list);

    return init_thunk_literal(list);
}

struct Thunk *makeInfinite(int start) {
    struct List *list = malloc(sizeof(struct List));

    list->content_type = INT;
    list->type = INFINITE;
    list->start = start;
    list->end = -1;
    list->last_eval = 0;
```

```
    list->curr_index = start;

    struct Thunk *data = makeInt(start);
    list->head = makeNode(data);


    list->last_eval = list->head;
    return init_thunk_literal(list);
}

struct Node *makeNode(struct Thunk *data_thunk) {
    struct Node *new = malloc(sizeof(struct Node));

    new->data = data_thunk;
    new->next = NULL;
    return new;
}


void explodeRangeList(void *list) {
    struct List *llist = (struct List *)list;


    while (llist->curr_index < llist->end) {
        evalNextNode(list);
    }
}

void evalNextNode(void *list) {
    struct List *llist = (struct List *)list;


    if (llist->type == RANGE || llist->type == INFINITE) {
        llist->curr_index++;
        struct Thunk *data = makeInt(llist->curr_index);
        struct Node *newNode = makeNode(data);
        llist = appendNode(llist, newNode);
    }
}

struct List *appendNode(struct List *list, struct Node *node) {
    if (!(list->head)) {
        list->head = node;
        list->last_eval = node;
    } else {
        (list->last_eval)->next = node;
        list->last_eval = node;
    }
    return list;
}
```

```
struct Thunk *appendNodeThunk(struct Thunk *list, struct Node *node) {
    return init_thunk_literal(appendNode(invoke(list),node));
}

void printAnyThunk(struct Thunk *primThunk, int *types, int index) {
    int ty = types[index];
    void *thing = invoke(primThunk);
    if (ty <= FLOAT) {
        printPrim(thing, ty);
    } else if (ty == LIST) {
        printList(thing, types, index);
    } else if (ty == TUPLE) {
        printTuple(thing, types, index);
    } else if (ty == MAYBE) {
        printMaybe(thing, types, index);
    }
}

void printList(struct List *list, int *types, int index) {
    struct Node *curr = list->head;
    int type = list->content_type;

    int nested_type_index = (2 * index) + 1;

        if (type != CHAR)
            printf("[");
        while (curr != NULL) {
                struct Thunk *ndata = curr->data;
                printAnyThunk(ndata, types, nested_type_index);
                curr = curr->next;
                if (curr && type != CHAR)
                    printf(", ");
        }
        if (type != CHAR)
            printf("]");
}

void printCompList(void *list);

void printPrim(void *data, int ty) {
    if (ty == INT) {
        int int_data = *(int *)data;
        printf("%d",int_data);
    } else if (ty == BOOL) {
        int bool_data = *(int *)data;
        if (bool_data) {
            printf("true");
        } else {
            printf("false");
        }
    } else if (ty == FLOAT) {
```

```
        float float_data = *(float *)data;
        printf("%f",float_data);
    } else if (ty == CHAR) {
        char char_data = *(char *)data;
        printf("%c",char_data);
    }
}

void printBool(char b) {
    printf("%s", b != 0 ? "true" : "false");
}


struct Thunk *makeIte(struct Thunk *cond_thunk, struct Thunk *then_thunk,
    struct Thunk *else_thunk){


    void *val = invoke(cond_thunk);
    char boolean_val = *(char *)(val);
    if(boolean_val){
        return then_thunk;
    } else {
    return else_thunk;
    }
}

void initNativeThunks() {
    // Integer operations
    init_thunk(add_init_thunk, &add_eval, 2, 0);
    init_thunk(sub_init_thunk, &sub_eval, 2, 0);
    init_thunk(mult_init_thunk, &mult_eval, 2, 0);
    init_thunk(divi_init_thunk, &divi_eval, 2, 0);
    init_thunk(mod_init_thunk, &mod_eval, 2, 0);
    init_thunk(powe_init_thunk, &powe_eval, 2, 0);
    init_thunk(eq_init_thunk, &eq_eval, 2, 0);
    init_thunk(neq_init_thunk, &neq_eval, 2, 0);
    init_thunk(geq_init_thunk, &geq_eval, 2, 0);
    init_thunk(leq_init_thunk, &leq_eval, 2, 0);
    init_thunk(powe_init_thunk, &powe_eval, 2, 0);
    init_thunk(eq_init_thunk, &eq_eval, 2, 0);
    init_thunk(neq_init_thunk, &neq_eval, 2, 0);
    init_thunk(geq_init_thunk, &geq_eval, 2, 0);
    init_thunk(less_init_thunk, &less_eval, 2, 0);
    init_thunk(greater_init_thunk, &greater_eval, 2, 0);
    init_thunk(neg_init_thunk, &neg_eval, 1, 0);

    // Float operations
    init_thunk(addf_init_thunk, &addf_eval, 2, 0);
    init_thunk(subf_init_thunk, &subf_eval, 2, 0);
    init_thunk(multf_init_thunk, &multf_eval, 2, 0);
    init_thunk(divf_init_thunk, &divf_eval, 2, 0);
```

```
    init_thunk(powef_init_thunk, &powef_eval, 2, 0);
    init_thunk(eqf_init_thunk, &eqf_eval, 2, 0);
    init_thunk(neqf_init_thunk, &neqf_eval, 2, 0);
    init_thunk(geqf_init_thunk, &geqf_eval, 2, 0);
    init_thunk(leqf_init_thunk, &leqf_eval, 2, 0);
    init_thunk(powef_init_thunk, &powef_eval, 2, 0);
    init_thunk(eqf_init_thunk, &eqf_eval, 2, 0);
    init_thunk(neqf_init_thunk, &neqf_eval, 2, 0);
    init_thunk(geqf_init_thunk, &geqf_eval, 2, 0);
    init_thunk(lessf_init_thunk, &lessf_eval, 2, 0);
    init_thunk(greaterf_init_thunk, &greaterf_eval, 2, 0);
    init_thunk(negf_init_thunk, &negf_eval, 1, 0);

    // Boolean operations
    init_thunk(andb_init_thunk, &andb_eval, 2, 0);
    init_thunk(orb_init_thunk, &orb_eval, 2, 0);
    init_thunk(notb_init_thunk, &notb_eval, 1, 0);

    // List operations
    init_thunk(cons_init_thunk, &cons_eval, 2, 0);
    init_thunk(cat_init_thunk, &cat_eval, 2, 0);
    init_thunk(length_init_thunk, &length_eval, 1, 0);
    init_thunk(head_init_thunk, &head_eval, 1, 0);
    init_thunk(tail_init_thunk, &tail_eval, 1, 0);

    // Tuple operations
    init_thunk(first_init_thunk, &first_eval, 1, 0);
    init_thunk(second_init_thunk, &second_eval, 1, 0);

    init_thunk(is_none_init_thunk, &is_none_eval, 1, 0);
    init_thunk(from_just_init_thunk, &from_just_eval, 1, 0);

    init_thunk(int_to_float_init_thunk, &int_to_float_eval,1, 0);

    init_thunk(ite_init_thunk, &ite_eval,3, 1);
}

void printTuple(void *tup, int *types, int index) {
    struct Tuple *t = tup;
    int nested_type_index1 = (2 * index) + 1;
    int nested_type_index2 = (2 * index) + 2;
    int t1 = types[nested_type_index1];
    int t2 = types[nested_type_index2];

    printf("(");
    printAnyThunk(t->first,types, nested_type_index1);
    printf(", ");
    printAnyThunk(t->second,types, nested_type_index2);
    printf(")");
}
```

```
void printMaybe(void *mayb, int *types, int index) {
    struct Maybe* m = mayb;
    int nested_type_index1 = (2 * index) + 2;
    m->ty = types[nested_type_index1];
    if (m->is_none) {
        printf("none");
    } else {
        printf("just ");
        printAnyThunk(m->data, types, nested_type_index1);
    }
}
```

8.23 codegen.ml (Da, Amanda, Hans, Hollis)

```
module L = Llvm
open Pretty_type_print
open Ast
open Tast
open Lib
open Structs
open Thunk
open Natives
open Mymap

module StringMap = Map.Make(String)

let translate (decl_lst: (decl * typed_decl) list) =
    let rec expon base pow = if pow = 0
        then 1
        else base * (expon base (pow - 1))
    in

    let rec get_type_depth = function
        | Int | Bool | Float | Char -> 1
        | TconList t -> 1 + get_type_depth t
        | Tmaybe t -> 1 + get_type_depth t
        | TconTuple(t1,t2) -> let d1 = get_type_depth t1 in
            let d2 = get_type_depth t2 in
            (if d2 > d1 then d2 else d1) + 1
        | _ -> 1
    in

    let rec throw_away_lambda = function
        | (TLambda(_, b), _) -> throw_away_lambda b
        | other -> other
    in


    (* Get non-lambda Vdefs and lambda vdefs*)
    let (var_lst, lambda_lst) =
        let islambda = function
            | (_, TypedVdef(_,(TLambda(_,_),_))) -> true
```

```
        | _ -> false
    in
    let notlambda = fun x -> not (islambda x)
    in
(List.filter (notlambda) decl_lst, List.filter (islambda) decl_lst)
in


(* fn to add terminal instruction if needed *)
let add_terminal builder instr =
    match L.block_terminator (L.insertion_block builder) with
          Some _ -> ()
        | None -> ignore (instr builder)
in

let get_ltyp tarr = match tarr with
      Tarrow(t1, t2) -> t1
    | _ -> raise (Failure "Not Tarrow")
in

let rec flatten_arrow_type = function
    | Tarrow(l,r) -> l :: (flatten_arrow_type r)
    | r -> [r]
in

let rec lambda_var_list = function
    | (TLambda(v,ex),_) -> v::(lambda_var_list ex)
    | _ -> []
in
let stack_alloc builder var argll =
    let stack_ref = L.build_alloca
        (L.pointer_type struct_thunk_type) var builder in
    let _ = L.build_store argll stack_ref builder in
    let loaded = L.build_load stack_ref "amanda" builder in
    loaded
in

(* convert Tlambda -> Tlambda_def *)
let tldef_convert (tlambda: typed_expr) (name: string) =
    match tlambda with
    | (TLambda(var, e), typ) ->
        {
            tlname = name;
            tltyp = (get_ltyp typ);
            trtyp = snd e;
            tlexp = (TVar var, get_ltyp typ);
            trexp = e;
        }
    | _ -> raise (Failure "not Tlambda")
in
```

```
(* list of tlambda_def's *)
let lm_defs: (tlambda_def list) =
    let to_lmdef (dec: decl * typed_decl) = match dec with
        | (_,TypedVdef(n,l)) -> tldef_convert l n
    in
List.map to_lmdef lambda_lst
in

(* get array of types of lambda *)
let rec arg_types (lmd: tlambda_def) =
    let first_arg = [| L.pointer_type struct_thunk_type |] in
    let rec get_args texp = match texp with
              (TLambda(_,txp), ty) ->
                  let typ = (L.pointer_type struct_thunk_type) in
                  Array.append [| typ |] (get_args txp)
            | _ -> [||]
    in
    let other_args = get_args lmd.trexp in
    Array.append first_arg other_args


in
let eval_decls: (L.llvalue * tlambda_def) StringMap.t =
    (* declare eval functions and put into a map*)
    let gen_decls m (lm_def: tlambda_def) =
        (*(* eval function: void *f(struct Thunk*) *)
        let eval_name = "$eval_" ^ lm_def.tlname in
        StringMap.add eval_name (L.define_function eval_name eval_func_type
             the_module, lm_def) m
in List.fold_left gen_decls StringMap.empty lm_defs
in

let fn_decls: (L.llvalue *tlambda_def) StringMap.t =
    let gen_decls m (lm_def: tlambda_def) =
        (* core function declaration *)
        let fname = lm_def.tlname in
        let fn_args = arg_types lm_def in


        (* core function: void *f(...)  *)
        let ftype = L.function_type (L.pointer_type i8_t) fn_args in
        StringMap.add fname (L.define_function fname ftype the_module,
            lm_def) m
in List.fold_left gen_decls StringMap.empty lm_defs
in

(* build thunk for each function and put into map *)


let build_thunk (lmd: tlambda_def) =
    (* get number of args of function *)
```

```
    let rec argnum texp = match texp with
          (TLambda(_,t),_) -> 1 + (argnum t)
        | _ -> 0
    in
    let argc = 1 + (argnum lmd.trexp) in
    let eval_name = "$eval_" ^ lmd.tlname in
    let eval_fn = fst (StringMap.find eval_name eval_decls) in
    let num_args = L.const_int i32_t argc in
    let thunk_name = "$$" ^ lmd.tlname in
    let is_ite = L.const_int i32_t 0 in

    let f_init_thunk = L.define_global (thunk_name ^ "_init_thunk")
        (L.const_null struct_thunk_type) the_module in

    let _ = L.build_call initThunk [| f_init_thunk ; eval_fn; num_args ;
    is_ite |]
        "initThunk" builder in ()


in

List.iter build_thunk lm_defs;

let build_eval_func_body eval_dcls = function
        | (_,TypedVdef(name,(txpr,Tarrow(l,r)))) ->
                let eval_decl = (match (StringMap.find_opt
                ("$eval_"^name) eval_dcls) with
                        | Some (decl,_) -> decl
                        | None -> raise (Failure ("No eval function for decl
                        "^name))) in
                let builder =
                        L.builder_at_end context (L.entry_block eval_decl) in


                let types = l :: (flatten_arrow_type r) in
                let num_args = List.length types in
                let ts = L.params eval_decl in
                let t = List.hd (Array.to_list ts) in

                (* allocate thunk, args, and returned val *)
                let p = L.build_alloca (L.pointer_type struct_thunk_type)
                "pthunk" builder in


                (* list of arg thunks to allocate *)
                let rec build_arg_alloca n = if n < 1 then [] else match n
                with
                        | 1 -> [(L.build_alloca (L.pointer_type
                        struct_thunk_type)
                        ("thunk1") builder)]
                        | m -> (L.build_alloca (L.pointer_type
```

```
        struct_thunk_type)
                ("thunk"^(string_of_int (m)))
                builder)::(build_arg_alloca (m-1))
in
let args = List.rev (build_arg_alloca (num_args-1))
in
let ret = L.build_alloca (L.pointer_type i8_t) "ret" builder
in

ignore(L.build_store t p builder);

(* load and dereference to get the args of the thunk *)
let load_store_arg a ind =
    let tload = L.build_load p "tload" builder in
    let args = L.build_gep tload
    [| L.const_int i32_t 0; L.const_int i32_t 3 |] "args"
    builder
    in
    let loadargs = L.build_load args "loadargs" builder in
    let arg = L.build_gep loadargs [|
        L.const_int i32_t ind |] "args" builder
    in
    let loadarg = L.build_load arg "loadarg" builder in
    ignore(L.build_store loadarg a builder)
in

let rec load_store_args ind lst = match lst with
    | [] -> ()
    | h::t -> load_store_arg h ind; load_store_args (ind+1) t
in

load_store_args 0 args;
(* load all args into list *)

let loaded_args: (L.llvalue list) =
    let rec load_arg arglst = match arglst with
        | [] -> []
        | h::t -> (L.build_load h "load" builder) ::
            (load_arg t)
in
load_arg args
in

(* call the function *)
let f = StringMap.find_opt name fn_decls in
let func = match f with
        | Some f -> fst f
        | None -> raise (Failure "function not found")
in
let result = L.build_call func (Array.of_list loaded_args)
"result" builder in
```

```
                (* store, load, cast, and return *)
                ignore(L.build_store result ret builder);
                let retload = L.build_load ret "retload" builder in
                let ret_cast = L.build_bitcast retload (L.pointer_type i8_t)
                "ret_cast" builder in
                L.build_ret ret_cast builder
        | (_,TypedVdef(name, _)) -> raise(Failure
        (name ^ "is not an arrow type!"))
in

let rec build_expr (texp: typed_expr) builder (scope: L.llvalue StringMap.t) =

    let ty_to_int = (function
        | Int -> 0
        | Float -> 1
        | Char -> 2
        | Bool -> 3
        | (TconList _) -> 4
        | (TconTuple _) -> 5
        | (Tmaybe _) -> 6
        |  _ -> -1)
    in

    let tex = fst texp in
    let typ = snd texp in match tex with
        (* literals - build thunk literals *)
          TIntLit n -> L.build_call makeInt [| L.const_int i32_t n |]
            "makeInt" builder
        | TFloatLit f -> L.build_call makeFloat [| L.const_float float_t f |]
            "makeFloat" builder
        | TCharLit c -> L.build_call makeChar [| L.const_int i8_t
        (Char.code c) |]
            "makeChar" builder
        | TBoolLit b -> L.build_call makeBool [| L.const_int i8_t
        (if b then 1 else 0) |]
            "makeBool" builder
        | TVar s -> (match (StringMap.find_opt s scope) with
              Some lval ->
                let p = L.build_alloca (L.pointer_type struct_thunk_type)
                "hans" builder in
                ignore(L.build_store lval p builder);
                let hans_load = L.build_load p "loaded_hans" builder in
                hans_load
            | None -> (match (L.lookup_global ("$$" ^ s ^ "_init_thunk")
            the_module ) with
                  Some l -> l
                | None -> (match (StringMap.find_opt s fn_decls) with
                      Some l_ -> fst l_
                    | None -> raise (Failure (s^ " not found in scope"))
                        ))
```

```
)
| TListRange(s,e) ->
    let start = build_expr s builder scope in
    let endd = build_expr e builder scope in
    L.build_call makeRangeList [| start ; endd |] "range" builder
| TListLit texlst ->
    let ty_code = match typ with
              TconList Int -> 0
            | TconList Bool -> 1
            | TconList Char -> 2
            | TconList Float -> 3
            | TconList (TconList _) -> 4
            | TconList (TconTuple _) -> 5
            | TconList (Tmaybe _) -> 6
            | ty -> -1
    in
    let emptylist = L.build_call makeEmptyList [| L.const_int i32_t
    ty_code |]
    "empty" builder in

    let rec build_list s prevlist = (match s with
        | h :: t ->
            let tn_star = build_expr h builder scope in
            let n_star = L.build_call makeNode [| tn_star |]
            "makeNode" builder in
            let nextlist = L.build_call appendNodeThunk
                [| prevlist ; n_star |] "appendNodeThunk" builder in
            build_list t nextlist
        | [] -> prevlist
    )
    in
    build_list texlst emptylist

| TLet (ta, t) -> (match ta with
                      TAssign (s, te) -> let v1 = build_expr te
                      builder scope in
                            let new_scope = StringMap.add s v1 scope
                            in
                            build_expr t builder new_scope
)
(* Application *)
| TApp(t1, t2) -> let lv1 = build_expr t1 builder scope in
    let lv2 = build_expr t2 builder scope in
    L.build_call apply [| lv1; lv2 |] "apply" builder

| TIte(cond, then_ex, else_ex) -> let cv =
    build_expr cond builder scope in
    let tv = build_expr then_ex builder scope in
    let ev = build_expr else_ex builder scope in
    let lv = L.build_call apply [| ite_init_thunk ; cv |] "apply" builder
    let lv1 = L.build_call apply [| lv ; tv |] "apply" builder in
```

```
      L.build_call apply [| lv1 ; ev |] "apply" builder


 | TAdd -> add_init_thunk
 | TSub -> sub_init_thunk
 | TMult -> mult_init_thunk
 | TDiv -> divi_init_thunk
 | TMod -> mod_init_thunk
 | TPow -> powe_init_thunk
 | TEq -> eq_init_thunk
 | TNeq -> neq_init_thunk
 | TGeq -> geq_init_thunk
 | TLeq -> leq_init_thunk
 | TLess -> less_init_thunk
 | TGreater -> greater_init_thunk
 | TNeg -> neg_init_thunk
 | TAddF -> addf_init_thunk
 | TSubF -> subf_init_thunk
 | TMultF -> multf_init_thunk
 | TDivF -> divf_init_thunk
 | TPowF -> powef_init_thunk
 | TEqF -> eqf_init_thunk
 | TNeqF -> neqf_init_thunk
 | TGeqF -> geqf_init_thunk
 | TLeqF -> leqf_init_thunk
 | TLessF -> lessf_init_thunk
 | TGreaterF -> greaterf_init_thunk
 | TNegF -> negf_init_thunk
 | TAnd -> andb_init_thunk
 | TOr -> orb_init_thunk
 | TNot -> notb_init_thunk
 | TCons -> cons_init_thunk
 | TCat -> cat_init_thunk
 | TLen -> length_init_thunk
 | THead -> head_init_thunk
 | TTail -> tail_init_thunk
 | TFirst -> first_init_thunk
 | TSec -> second_init_thunk
 | TIs_none -> is_none_init_thunk
 | TFrom_just -> from_just_init_thunk
(* | TFrom_just -> from_just_init_thunk*)
 | TInt_to_float -> int_to_float_init_thunk
 | TLambda(_, _) -> raise(Failure "unexpected lambda")
 | TTuple(tx1,tx2) ->
        let (t1,t2) = (match typ with
                | (TconTuple l) -> l
                | _ -> raise(Failure "expected TconTuple")) in
        let first = build_expr tx1 builder scope in
        let sec = build_expr tx2 builder scope in
        L.build_call makeTuple
        [| first ; sec
```

```
        ; L.const_int i32_t (ty_to_int t1)
        ; L.const_int i32_t (ty_to_int t2) |] "tup" builder
| TJust(tx) -> let inn = build_expr tx builder scope in
        let t = (match typ with
                | (Tmaybe l) -> l
                | _ -> raise (Failure "expected Tmaybe")) in
        L.build_call makeMaybe
        [| inn ; L.const_int i32_t (ty_to_int t) |] "just" builder
| TNone ->
        let t = (match typ with
                | (Tmaybe l) -> l
                | _ -> raise (Failure "expected Tmaybe")) in
        L.build_call makeMaybe
        [| L.const_null (L.pointer_type struct_thunk_type)
        ; L.const_int i32_t (ty_to_int t) |] "none" builder

| TListComp (tex, tclslst) ->
        let ty_code = match typ with
                     TconList Int -> 0
                   | TconList Bool -> 1
                   | TconList Char -> 2
                   | TconList Float -> 3
                   | TconList (TconList _) -> 4
                   | TconList (TconTuple _) -> 5
                   | TconList (Tmaybe _) -> 6
                   | ty -> -1
        in
        (*let num_vars =
            let rec num_list lst = (match lst with
                | h::t -> (match h with
                        | TListVBind _ -> 1 + num_list t
                        | _ -> num_list t
                )
                | [] -> 0
            )
        in num_list tclslst in*)

        let rec get_lists clauses = match clauses with
            | h::t -> (match h with
                | TListVBind (str, texp) ->
                    let b = build_expr texp builder scope in
                        (str, b)::(get_lists t)
                | _ -> get_lists t
            )
            | [] -> []
        in
        let rec get_filters clauses = match clauses with
            | h::t -> (match h with
                | TFilter texp ->
                    let b = build_expr texp builder scope in
                        b::(get_filters t)
```

```
                        | _ -> get_filters t
                    )
                    | [] -> []
                in
                let lists = get_lists tclslst in
                let filters = get_filters tclslst
                in


                let fn_thunk = build_expr tex builder scope in
                let map_thunk = L.build_call mapl
                    [| snd (List.hd lists); fn_thunk; L.const_int i32_t
                    ty_code |] "map_thunk" builder in

                let rec apply_map_list lsts thunk = match lsts with
                    | [] -> thunk
                    | h::t -> let map_list_thunk = L.build_call map_listl
                            [| thunk; snd (List.hd lsts) ; L.const_int i32_t
                            ty_code |]
                            "map_list_thunk" builder in
                             apply_map_list t map_list_thunk
                in
                let mapped_thunk = apply_map_list (List.tl lists) map_thunk
                in
                let rec apply_filters fltrs thunk = match fltrs with
                    | [] -> thunk
                    | h::t -> let filter_thunk = L.build_call filterl
                            [| thunk; List.hd fltrs ; L.const_int i32_t
                            ty_code |] "filter_thunk"
                            builder
                            in apply_filters t filter_thunk
                in
                apply_filters filters mapped_thunk

        | _ -> raise(Failure "couldn't build texpr")
in


(* GIVE THIS fn_decls *)
let build_func_body func_decls = function
    | (_,TypedVdef(name,(txpr,Tarrow(l,r)))) ->
        let fn_decl = (match(StringMap.find_opt name func_decls) with
            | Some (decl,_) -> decl
            | None -> raise (Failure ("No function for decl "^name))) in
        let fn_builder = L.builder_at_end context (L.entry_block fn_decl) in
        let vars = lambda_var_list (txpr,Tarrow(l,r)) in
        let argsll = L.params fn_decl in
        let argslll = Array.to_list argsll in
        let var_to_argsll_map = List.fold_left2 (fun map var ll ->
            StringMap.add var ll map) StringMap.empty vars argslll in
        let var_to_local_map =
```

```
            StringMap.mapi (stack_alloc fn_builder) var_to_argsll_map in
        let lbody = throw_away_lambda (txpr, Tarrow(l,r)) in
        let l_body_expr = build_expr lbody fn_builder var_to_local_map in
        let bc = L.build_call invoke [| l_body_expr |] "da" fn_builder in
        add_terminal fn_builder (L.build_ret bc)
    | _ -> raise(Failure "expected tarrow vdef 0_o")
in


let print_expr (lv: L.llvalue) (vtype: ty) ty_array =
    (* call invoke on thunk *)
    let _ = L.build_call invoke [| lv |] "invoke" builder in
    (* print *)
    let _ = (match vtype with
        | TconList(t) -> L.build_call printAnyThunk
            [| lv ; ty_array ; L.const_int i32_t 0 |] "" builder
        | Int -> L.build_call printAnyThunk
            [| lv ; ty_array ; L.const_int i32_t 0 |] "" builder
        | Bool -> L.build_call printAnyThunk
            [| lv ; ty_array ; L.const_int i32_t 0 |] "" builder
        | Float -> L.build_call printAnyThunk
            [| lv ; ty_array ; L.const_int i32_t 0 |] "" builder
        | Char -> L.build_call printAnyThunk
            [| lv ; ty_array ; L.const_int i32_t 0 |] "" builder
        | TconTuple _ -> L.build_call printAnyThunk
            [| lv ; ty_array ; L.const_int i32_t 0 |] "" builder
        | Tmaybe _ -> L.build_call printAnyThunk
            [| lv ; ty_array ; L.const_int i32_t 0 |] "" builder
        | ty -> raise(Failure("Main is of unprintable type"^(ty_to_str ty)))
    ) in
    L.build_call printf_func [| char_format_str ; L.const_int i8_t
    (Char.code('\n')) |] "printf" builder


in

let rec write_type_array i ty arr = match ty with
    | Int ->
        let pos = L.build_gep arr [| L.const_int i32_t i |] "pos" builder in
        ignore (L.build_store (L.const_int i32_t 0) pos builder)
    | Bool ->
        let pos = L.build_gep arr [| L.const_int i32_t i |] "pos" builder in
        ignore (L.build_store (L.const_int i32_t 1) pos builder)
    | Char ->
        let pos = L.build_gep arr [| L.const_int i32_t i |] "pos" builder in
        ignore (L.build_store (L.const_int i32_t 2) pos builder)
    | Float ->
        let pos = L.build_gep arr [| L.const_int i32_t i |] "pos" builder in
        ignore (L.build_store (L.const_int i32_t 3) pos builder)
    | TconList (inner_ty) ->
        let pos = L.build_gep arr [| L.const_int i32_t i |] "pos" builder in
```

```
            ignore (L.build_store (L.const_int i32_t 4) pos builder);
            write_type_array (2 * i + 1) inner_ty arr
        | TconTuple (inner_ty1, inner_ty2) ->
            let pos = L.build_gep arr [| L.const_int i32_t i |] "pos" builder in
            ignore (L.build_store (L.const_int i32_t 5) pos builder);
            write_type_array (2 * i + 1) inner_ty1 arr;
            write_type_array (2 * i + 2) inner_ty2 arr
        | Tmaybe (inner_ty) ->
            let pos = L.build_gep arr [| L.const_int i32_t i |] "pos" builder in
            ignore (L.build_store (L.const_int i32_t 6) pos builder);
            write_type_array (2 * i + 2) inner_ty arr
        | _ -> raise (Failure "main should be a concrete non-arrow type!")

    in

    let rec build_decl (tdecl: (decl * typed_decl)) =
        match tdecl with
            | (_, TypedVdef("main",texp)) ->
                let type_depth = get_type_depth (snd texp) in
                let ty_heap = L.build_array_alloca i32_t
                    (L.const_int i32_t ((expon 2 (type_depth))-1 ) )
                    "ty_heap" builder in
                write_type_array 0 (snd texp) ty_heap;
                let v = build_expr texp builder StringMap.empty in
                ignore (print_expr v (snd texp) ty_heap)
            | (_, TypedVdef(name,(tex,Tarrow(_)))) as tup->
                let _ = build_eval_func_body eval_decls tup in
                build_func_body fn_decls tup
                (* build_func_body *)
            | (_, TypedVdef(name, _)) -> raise(Failure (name ^ " is not an arrow
            type!"))
    in
    let _ = List.iter build_decl decl_lst in
        ignore (L.build_ret (L.const_int i32_t 0) builder);
    the_module

get_fresh_var.ml (Hans)
let counter = ref 0

let get_fresh a =
    let var_string = a ^ string_of_int !counter in
    counter := !counter + 1;
    var_string
```

8.24 pretty_type_print.ml (Amanda, Da)

```
open Ast

let rec ast_to_str exp =
    match exp with
    | Ite(e1,e2,e3) -> "if (" ^ (ast_to_str e1) ^
                       ")\nthen (" ^ (ast_to_str e2) ^
```

```
                  ")\nelse(" ^ (ast_to_str e3) ^ ")"
    | Let(e1,e2) -> "let (" ^ (assign_to_str e1) ^
                  ")\nin (" ^ (ast_to_str e2) ^ ")"
    | Lambda(e1,e2) -> "fun " ^ (e1) ^ " -> \n(" ^ (ast_to_str e2) ^ ")"
    | App(App(op, arg1), arg2) -> "((" ^ op_to_str op ^ ")~"
        ^ (ast_to_str arg1) ^ ")~" ^ (ast_to_str arg2)
    | App(op, e) ->  "(" ^ (op_to_str op) ^ ")~" ^ (ast_to_str e)
    | Var(s) -> s
    | Tuple(a,b) -> "("^(ast_to_str a)^", "^(ast_to_str b)^")"
    (* Lists *)
    | ListLit((Ast.CharLit c) :: tl) ->
        "\"" ^ (char_list_to_str ((Ast.CharLit c) :: tl)) ^ "\""
    | Just(e) -> "just "^(ast_to_str e)
    | ListLit(alist) -> "[" ^ (list_to_str alist) ^ "]"
    | ListRange(e1, e2) -> "[" ^ (ast_to_str e1) ^ "..." ^ (ast_to_str e2) ^ "]"
    | ListComp(e, c) -> "[" ^ (ast_to_str e) ^ "|" ^ (clauses_to_str c) ^ "]"
    | None -> "none"
    | BoolLit(b) -> string_of_bool b
    | CharLit(c) -> String.make 1 c
    | IntLit n -> string_of_int n
    | FloatLit f -> string_of_float f
    | WildCard -> "_"
    | _ -> ""

and assign_to_str = function | Assign (e1,e2)
    -> (e1) ^ "=" ^ (ast_to_str e2)

and clauses_to_str clauses =
   match clauses with
   | [Filter(e)] -> "Filter(" ^ (ast_to_str e) ^"), "
   | [ListVBind(s, l)] -> "(" ^ s ^ " over " ^ "[" ^ (ast_to_str l) ^ "]), "
   | h::t -> (clauses_to_str [h]) ^ clauses_to_str t
   | [] -> ""

and op_to_str s =
    match s with
    (* Boolean Operators *)
    | Or -> "or"
    | And -> "and"
    | Not -> "not"
    | Eq -> "=="
    | EqF -> "==."
    | Neq -> "!="
    | NeqF -> "!=."
    | Less -> "<"
    | LessF -> "<."
    | Greater -> ">"
    | GreaterF -> ">."
    | Leq -> "<="
    | LeqF -> "<=."
    | Geq -> ">="
```

```
    | GeqF -> ">=."
    (* Math Operations *)
    | Add -> "+"
    | Sub -> "-"
    | Mult -> "*"
    | Div -> "/"
    | AddF -> "+."
    | SubF -> "-."
    | MultF -> "*."
    | DivF -> "/."
    | Pow -> "^"
    | PowF -> "^."
    | Neg -> "neg"
    (* List Operations *)
    | Cons -> "cons"
    | Cat -> "cat"
    | Head -> "head"
    | Tail -> "tail"
    | Len -> "len"
    | First -> "first"
    | Sec -> "sec"
    | Is_none -> "is_none"
    | From_just -> "from_just"
    | Int_to_float -> "int_to_float"
    | _ -> ast_to_str s

and ty_to_str ty =
    match ty with
    | Bool -> "bool"
    | Int -> "int"
    | Char -> "char"
    | Float -> "float"
    | TconList(t) -> "["^ (ty_to_str t) ^ "]"
    | TconTuple(t1,t2) -> "(" ^ (ty_to_str t1) ^ "," ^ (ty_to_str t2) ^ ")"
    | Tmaybe(t) -> "maybe " ^ (ty_to_str t)
    | Tvar(t) -> t
    | Tarrow(t1,t2) -> (nestarrow t1) ^ " -> " ^ (ty_to_str t2)
    | Tforall(_,t) -> ty_to_str t

and nestarrow ty =
    match ty with
    | Tarrow(t,t1) -> "("^(nestarrow t)^" -> "^(ty_to_str t1)^")"
    | el -> ty_to_str el


and char_list_to_str cl =
    match cl with
    | [] -> ""
    | [Ast.CharLit c] -> String.make 1 c
    | charlist -> let convert x = match x with
        | Ast.CharLit c -> c
```

```
        | _ -> 'F'
        in List.fold_left (^) "" (List.map (String.make 1) (List.map convert
        charlist))

and list_to_str el = match el with
        | [] -> ""
        | hd::tl ->  List.fold_left (fun curr_str e -> curr_str ^ (ast_to_str e) ^
        ",") "" el
let rec print_annot_pairs lst = match lst with
    | (Annot(n1, t), Vdef(n2, e)) :: tl ->
        print_endline ("a_name: " ^ n1 ^ ", v_name: " ^ n2 ^ ", type:" ^
        (ty_to_str t));
        print_annot_pairs tl
    | [] -> raise(Failure "empty program")
    | _ -> raise(Failure "expected annot vdef pairs")
```

8.25 pretty_tast_print.ml (Amanda, Hollis)

```
open Ast
open Tast
open Pretty_type_print

let rec texpr_to_str tx =
    match tx with
    | TIntLit (i) -> string_of_int i
    | TFloatLit (f) -> string_of_float f
    | TBoolLit (b) -> string_of_bool b
    | TCharLit (c) -> String.make 1 c
    | TWildCard -> "_"
    | _ -> "texpr"

let typ_to_str ty =
    Pretty_type_print.ty_to_str ty

let rec tast_to_str tdec =
    match tdec with
    | (TypedVdef (name, (texpr1, infty))) -> "TypedVdef(" ^ (name) ^ ", (" ^
    (texpr_to_str texpr1) ^ ", " ^ (typ_to_str infty) ^ "))"
```

8.26 compiler.ml (Amanda)

```
open Ast
open Tast
open Scanner
open Parser
open Check_main
open Pair_annots
open Pretty_type_print
open Lift_lambdas
open Check_lists
open Printf
open Remove_substs
```

```
open Sys
open String
open Thunk
open Unix
module StringSet = Set.Make(String)

let print_decls d = match d with
        | Vdef(n, e) -> print_endline (n ^ " = "
            ^ (Pretty_type_print.ast_to_str (e)) ^ "\n");
        | Annot(s, t) -> print_endline (s ^ " :: "
            ^ (Pretty_type_print.ty_to_str (t)) ^ "\n")

let rec remove_path str =
    let slash = index_opt str '/' in
    match slash with
    | (Some i) -> remove_path (sub str (i+1) ((length str) -i-1))
    | None -> str

let print_tdecl td =
    match td with
    | (annot, tvd) -> print_endline ( Pretty_tast_print.tast_to_str tvd)

let rec print_all_types = function
    | (_, TypedVdef(name, (_,ty)))::xs ->
        print_endline (name ^ ": " ^ (ty_to_str ty));
        print_all_types xs
    | [] -> print_newline ()

let read_full_file fname =
            let ch = open_in fname in
            let s = really_input_string ch (in_channel_length ch) in
            close_in ch;
            s

let collect_args =
    let num_args = Array.length argv in
    let rec collect_args_n = function
        | 0 -> []
        | n ->  let argn = argv.(n) in
                let len = length argn in
                if len > 2 then
                if (sub argn (len-3) 3) = "rpl"
                then argn::(collect_args_n (n-1))
                else (collect_args_n (n-1)) @ [argn]
                else (collect_args_n (n-1)) @ [argn]
    in collect_args_n (num_args-1)

let rec parse_flags = function
    | (flag::xs) -> let (t,l,p) = parse_flags xs in
        if flag = "-t"
        then (true,l,p)
```

```
        else if flag = "-l"
        then (t,true,p)
        else if flag = "-p"
        then (t,l,true)
        else raise (Failure "Usage: <.rpl file> [-t, -l, -p]")
    | [] -> (false, false, false)

let _ =
    let all_args = collect_args in
    let input = List.hd all_args in
    let length = length input in
    (
    if length > 4
        then ()
        else raise (Failure "Usage: <.rpl file> [-t, -l, -p]")
    );
    let base = sub input 0 (length-4) in
    let base_no_path = remove_path base in
    let extension = sub input (length-3) 3 in

    let (printTypes,printLifted,printDecl) = parse_flags (List.tl all_args )in

    let _ = if extension = "rpl"
            then ()
            else raise (Failure "Usage: <.rpl file> [-t, -l, -p]") in

    let file_contents = read_full_file input in


    let lexbuf = Lexing.from_string file_contents in
    let program = Parser.program Scanner.token lexbuf in
    (if printDecl && (not printLifted)
        then List.iter print_decls program
        else ());
    let m_program = Lift_lambdas.transform_main program false in
    let program_ll = Lift_lambdas.close_and_lift m_program in
    (if printLifted
        then List.iter print_decls program_ll
        else ());
    let pair_program = Pair_annots.pair_av program_ll in
    let (subst,pair_iprogram) = Type_inference.type_paired_program pair_program in
    let pair_tprogram = Remove_substs.remove_subst_pairs subst pair_iprogram in
    (if printTypes
        then print_all_types pair_tprogram
        else ());
    let m = Codegen.translate pair_tprogram in
        Llvm_analysis.assert_valid_module m;
    let ls = Llvm.string_of_llmodule m in
    let file = base_no_path ^ ".byte" in
    let home = Unix.getenv "HOME" in
        if (not printTypes && not printLifted) then (let oc = open_out file
```

```
in
fprintf oc "%s\n" ls;
        close_out oc;
if (command ("llc -relocation-model=pic " ^ file) != 0)
    then raise (Failure "llc: non-zero exit code")
else if (command
    ("gcc -L"^home^"/rippl/src "
    ^ file ^".s -lall -lm  -o"
    ^ base_no_path ) != 0)
    then raise (Failure "gcc: non-zero exit code")
else ()) else ()
```

9 Sample Programs

9.1 Mandelbrot

The Mandelbrot set is constructed on the imaginary plane that iterates starting from 0 over the rule:

$$z_{n+1} = z_n^2 + c$$

Here z_0 is always the origin and c is the point over which we are iterating in the complex plane. A point is considered in the Mandelbrot set if it converges to the origin without escaping a radius of 2. Points that escape and are not in the Mandelbrot set are colored and ASCII encoded based on the number of iterations it takes for it to escape.

```
normSquared :: float -> float -> float
normSquared = fun a -> fun b-> a ^. 2.0 +. b ^. 2.0

encode :: int -> char
encode = fun count ->
    if count == 0
        then 'D'
        else if count < 10
            then 'C'
            else if count < 50
                then 'B'
                else if count < 200
                    then 'A'
                    else ' '


iterateMandelbrot :: int -> float -> float -> float -> float -> char
iterateMandelbrot = fun c -> fun b_z -> fun a_z -> fun b -> fun a ->
    if (c > 200 )
        then (encode~c)
        else if ((normSquared~(a_z *. a_z -. b_z *. b_z +. a)~
                (2.0 *. a_z *. a_z +. b)) >. 4.0)
            then (encode~c)
            else (iterateMandelbrot~(c+1)~(2.0*.a_z*.b_z +. b)~
                (a_z*.a_z -. b_z*.b_z +. a)~b~a)
```

```
everyOther :: [a] -> [a]
everyOther = fun list -> if len list <= 1
                    then list
                    else (head list) cons (everyOther~(tail (tail list)))

iterateReal :: float -> [char]
iterateReal = fun b ->
    let range = [(int_to_float x) /. 110.0 | x over [-220...110]] in
    let realRange = everyOther~range in
    let realLine = [(iterateMandelbrot~0~0~0~b~p) | p over realRange] in
    realLine cat ['\n']

main :: [char]
main =
    let range = [(int_to_float x) /. 50.0 | x over [-50...50]] in
    let complexRange = everyOther~range in
    let iterComplexRec = fun r ->
        (if len r == 0
            then " "
            else (iterateReal~(head r)) cat (iterComplexRec~(tail r)) )
    in iterComplexRec~complexRange
```

9.2 Fibonacci

The Fibonacci sequence is a sequence beginning with 0 and 1 with each number being the sum of the the two preceding it.

```
fib = fun n -> if n < 0 then 0
        else if n <= 1 then 1
        else fib~(n-1) + fib~(n-2)
main :: int
main = fib~6
```