1 Map Function

Pseudo-Wyvern

```
def map(f: A →φ B, l: List[A]): List[B] with φ =
  if isnil l then []
  else cons (f (head l)) (map (tail l f))
```

λ-Calculus

```
map = λφ. λA. λB.
  λf: A→φB.
    (fix (λmap: List[A] → List[B]).
      λl: List[A].
        if isnil l then []
        else cons (f (head l)) (map (tail l f)))
```

Typing

- This has the type: $\forall\phi.\forall A.\forall B.(A \to_\phi B) \to_\varnothing \texttt{List}[A] \to_\phi \texttt{List}[B]$ `with` $\varnothing$.
- `map` $\varnothing$ is a pure version of map.
- `map` $\{$`File`.$*\}$ is a version of map which can perform operations on `File`.

2 Dependency Injection

Pseudo-Wyvern

An HTTPServer module provides a single `init` method which returns a `Server` that responds to HTTP requests on the supplied socket.

```
module HTTPServer

def init(out: A <: {File, Socket}): Str →A.write Unit with ∅ =
  λ msg: Str.
    if (msg == ‘‘POST’’) then out.write(‘‘post response’’)
    else if (msg == ‘‘GET’’) then out.write(‘‘get response’’)
    else out.write(‘‘client error 400’’)
```

The main module calls `HTTPServer.init` with the `Socket` it should be writing to.

```
module Main
require HTTPServer, Socket

def main(): Unit =
  HTTPServer.init(Socket) ‘‘GET /index.html’’
```

The testing module calls `HTTPServer.init` with a `LogFile`, perhaps so the responses of the server can be tested offline.

```
module Testing
require HTTPServer, LogFile

def testSocket(): =
  HTTPServer.init(LogFile) ‘‘GET /index.html’’
```

λ-Calculus

The HTTPServer module:

```
1 HTTPServer = λx: Unit.
2    λA <: {File, Socket}.
3       λout: A.
4          λmsg: Str. A.write
```

The Main module:

```
1 Main = λhs: HTTPServer. λsock: Socket.
2    λx: Unit.
3       (hs sock) ‘‘GET /index.html’’
```

The Testing module:

```
1 Testing = λhs: HTTPServer. λlf: LogFile.
2    λx: Unit.
3       (hs lf) ‘‘GET /index.html’’
```

Types

– `HTTPServer.init` has the type $\lambda A <: \{\texttt{File}, \texttt{Socket}\}.\ A \to_{\varnothing} \texttt{Str} \to_{A.write} \texttt{Unit}$