“” Lisp Lisp  first-class functionLisp “” Lisp “”pure

“”`random`“” `random()`

“” Haskell“” Haskell “”side-effect……

Haskell  :p

“”Haskell

Haskell  IRC Haskell  Haskell :p

Haskell “” Haskell “”“”“”“”“”Lisp “”

Haskell  Pascal `x:=1`C  Java `x=1` Scheme `(set! x 1)`Common Lisp `(setq x 1)`“”state“”“”“”“” C `random()`

```
int random()
{
  static int seed = 0;
  seed = next_random(seed);
  return seed;
}
```

`seed` “static ” `random`  `random()` `next_random(seed)`  `seed`  `random()` `seed`  `random()` `seed`  `seed` `random()`

Haskell  Haskell “” Haskell  `random` “” `random` “”Haskell “” `random`  Haskell  `random` “”

```
--->
```

`random` “”“”“”“”

“”“” `random()`

“”Haskell  monad “”overloading Haskell “” monad “” monad transformer  monad transformer  hack monad  monad transformer

monad

monad “”

“”“”monad Haskell

“” Haskell “” Lisp “” C

C “”

```
int f(int x) {
    int y = 0;
    int z = 0;
    y = 2 * x;
    z = y + 1;
    return z / 3;
}
```

`f(x) = (2x+1)/3 y  z` “”“”“”“”“”

`y  z`“”“”“”“”“”

```
int f(int x) {
    return g(2 * x);
}

int g(int y) {
    return h(y + 1);
}

int h(int z) {
    return z/3;
}
```

f y z “”“”

“” HaskellML “”CMU  Robert Harper “”“”“”“”“”

X“” X “”90%“”“”“”“”

“”∀ Church  Church“”

“”“”

“”“”“”“”“” Lisp  ML “”