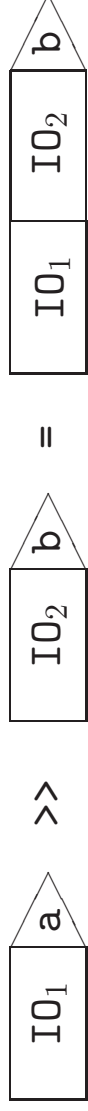


Functions Operating on the IO Monad

`putChar :: Char -> IO ()` `getChar :: IO Char`

`(>>) :: IO a -> IO b -> IO b` `return :: a -> IO a`



`(>>=) :: IO a -> (a -> IO b) -> IO b`



`getChar >> return ()` reads a character and ignores it

`getChar >>= putChar` reads a character and prints it on the screen

do-Notation

```
gets :: Int -> IO String
gets n = if n <= 0
  then return []
  else getChar >>= \x ->
    gets (n-1) >>= \xs ->
      return (x:xs)
```

```
p >>= \x ->
  q >>= \y ->
    r
do x <- p
  y <- q
  r
```

can be written as

```
gets :: Int -> IO String
gets n = if n <= 0 then return []
  else do x <- getChar
    xs <- gets (n-1)
    return (x:xs)
```