G51PGP Programming Paradigms
Case Study 2
String Transmitter

Abstract

The goal of this case study is to write a Haskell script that simulates the transmission of a string of characters as a list of bits.

Type Definition

Your script must contain the following type definition:

```
type Bit = Int
```

That is, a bit (either 0 or 1) is represented as an integer.

Important note: for this case study, binary numbers (lists of bits) are stored in the *reverse* order to normal, as this simplifies the definition of some functions on such lists. For example, the list `[1,0,1,1]` represents the binary number 1101, which in turn represents the natural number 13.

Function Definitions

• **Exercise:** Define a function

```
tobin :: Int -> [Bit]
```

that converts a natural number into the corresponding binary number. For example, `tobin 13` should give the binary number `[1,0,1,1]`.

• **Exercise:** Define a function

```
make8 :: [Bit] -> [Bit]
```

that makes a binary number into an 8-bit binary number, by chopping off extra bits from the end if there are more than 8 bits, or adding extra 0s to the end if there are less than 8 bits. For example, `make8 [1,0,1,1]` should give the binary number `[1,0,1,1,0,0,0,0]`.

• **Exercise:** Define a function

```
encode :: String -> [Bit]
```

that converts a string of characters into a list of bits by converting each character into a binary number (the library function `ord :: Char -> Int` converts a character into a natural number), making each binary number into an 8-bit number, and concatenating these numbers together. For example, `encode "Hugs"` should give the following list of bits:

```
[0,0,0,1,0,0,1,0,
 1,0,1,0,1,1,1,0,
 1,1,1,0,0,1,1,0,
 1,1,0,0,1,1,1,0]
```

• **Exercise:** Define a function

```
frombin :: [Bit] -> Int
```

that converts a binary number into the corresponding natural number. For example, `frombin [1,0,1,1]` should give the natural number 13.

• **Exercise:** Define a function

```
chop8 :: [Bit] -> [[Bit]]
```

that chops up a list of bits into a list of 8-bit binary numbers.

• **Exercise:** Define a function

```
decode :: [Bit] -> String
```

that converts a list of bits into a string of characters by chopping the list into 8-bit binary numbers, and converting these numbers into characters (`chr :: Int -> Char` converts a natural number into a character.)

• **Exercise:** Define a function

```
send :: String -> String
```

that simulates the transmission of a string of characters by encoding the string as a list of bits, and then decoding the resulting list as a string. For example, `send "Haskell"` should give `"Haskell"`.

— The End —