2.71

Problem

We want to create a data structure where 4 signed bytes are packed into a 32-bit unsigned int (`typedef unsigned packed_t`). We need to be able to extract a desired byte from the packed "word".

Visualization

```
1100 0011 1010 0101 0011 1100 0101 1010  <- unsigned int
+---+---+ +---+---+ +---+---+ +---+---+
    |         |         |         |
 Byte 3    Byte 2    Byte 1    Byte 0
```

Answers

A. Faulty implementation

The code is trying to bit shift the word in such a way that the desired byte is located at the last 8 bits of the shifted word.

```
int xbyte(packed_t word, int bytenum) {
  int shifted_word = (word >> (bytenum << 3));  // shift word by (bytenum * 8)
  return (shifted_word & 0xFF);  // extract only last 8 bits of shifted_word
}
```

The issue lies in how the last 8 bits are extracted. Since the shifted word is always bitwise and'ed with 0xFF, the padding is always filled with zeros. This is undesired since our byte should represent a *signed* int, which if negative, should be padded by ones.

B. Correct implementation

```
int xbyte(packed_t word, int bytenum) {
  // shift word so that byte occupies the 1st 8 bits (leftmost side)
  int left_shifted_word = (word << (3 - (bytenum << 3)));

  // shift byte to rightmost side (padded by ones if neg. else zeros)
  return (left_shifted_word >> 24);
}
```

2.82

Problem

The following is the initial code for the problem:

```c
// ints are 32 bits for this problem
int x = random();
int y = random();

// unsigned ints are 32 bits for this problem
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
```

Given the expressions for each problem, determine if they are *always* true. If yes, show a mathematical proof, else give a counterexample.

Answers

A. `(x < y) == (-x > -y)` False, if `x` is *TMIN*, then the first expression is always true for any value of `y` that is greater than *TMIN*. However, in that case, `-x` is *TMIN*, which any value of `-y` cannot be less than.

For example:

x	y	(x < y)	(-x > -y)
TMIN	0	1	0

B. `((x + y) << 4) + (y - x) == 17*y + 15*x` True. If one side overflows the other overflows as well.

$$\begin{aligned} ((x+y) << 4) + (y-x) &= 17y + 15x \\ (x+y) << 4 &= 16y + 16x \\ &= 16(y+x) \\ &= 2^4(y+x) \end{aligned}$$

C. `~x + ~y + 1 == ~(x + y)` True, by using definition of negation: $-x = \sim x + 1$.

$$\begin{aligned} \sim x + \sim y + 1 &= (-x-1) + (-y-1) + 1 \\ &= -(x+y) - 1 \\ &= \sim (x+y) \end{aligned}$$

D. **`(ux - uy) == -(unsigned) (y - x)`** True. The explicit cast serves no purpose because it only alters the interpretation of the bits. Using the definition of negation, the result of negating `(unsigned) (y - x)` and `(y - x)` would be the same. Therefore, we can distribute in the negative.

$$\begin{aligned} -(\text{unsigned})(y - x) &= -(y - x) && \text{(same bit pattern)} \\ &= x - y \end{aligned}$$

Since we are doing a comparison, both sides are interpreted as unsigned, therefore

$$\begin{aligned} ux - uy = x - y \\ = -(\text{unsigned})(y - x) \end{aligned}$$

E. **`((x >> 2) << 2) <= x`** True

$$\begin{aligned} ((x >> 2) << 2) &= \text{floor}(x/4) \cdot 4 \\ &\leq (x/4) \cdot 4 \\ &= x \end{aligned}$$

2.83

Problem

Consider binary numbers with a representation of the form $0.y\ y\ y\ y\ldots$ where y is a k-bit sequence. Let Y represent the decimal value for y.

For example:

1. 1/3 -> 0.01010101... (y = 01, k = 2, Y = 1)
2. 1/5 -> 0.00110011... (y = 0011, k = 4, Y = 3)

Answers

A. Find the expression for the value of these binary numbers in terms of Y and k

The expression to find the base-10 value of the given form of a binary number is the following:

$$\sum_{n=1}^{\infty} Y(\frac{1}{2^k})^n$$

This can be written as a infinite geometric series, which has the following property:

$$\sum_{n=0}^{\infty} ar^n = \frac{a}{1-r}$$

Re-arranging our equation so that we can exploit this, we get:

$$\left[\sum_{n=0}^{\infty} Y(\frac{1}{2^k})^n\right] - Y$$

Finally, we get:

$$\frac{Y}{1-\frac{1}{2^k}} - Y = \left(\frac{Y}{2^k - 1}\right)$$

B. Numeric Values of Examples

(a) y = 101

- k = 3
- Y = 5
- **numeric value =** $\frac{5}{6}$

(b) y = 0110

- k = 4
- Y = 6
- **numeric value =** $\frac{6}{15}$

(c) y = 010011

- k = 6
- Y = 19
- **numeric value =** $\frac{19}{65}$