```
    /* Get the sign bits */
    unsigned sx = ux >> 31;
    unsigned sy = uy >> 31;

    /* Give an expression using only ux, uy, sx, and sy */
    return ________ ;
}
```

2.84 ◆

Given a floating-point format with a k-bit exponent and an n-bit fraction, write formulas for the exponent E, significand M, the fraction f, and the value V for the quantities that follow. In addition, describe the bit representation.

A. The number 7.0

B. The largest odd integer that can be represented exactly

C. The reciprocal of the smallest positive normalized value

2.85 ◆

Intel-compatible processors also support an "extended precision" floating-point format with an 80-bit word divided into a sign bit, $k = 15$ exponent bits, a single *integer* bit, and $n = 63$ fraction bits. The integer bit is an explicit copy of the implied bit in the IEEE floating-point representation. That is, it equals 1 for normalized values and 0 for denormalized values. Fill in the following table giving the approximate values of some "interesting" numbers in this format:

	Extended precision	
Description	Value	Decimal
Smallest positive denormalized	________	________
Smallest positive normalized	________	________
Largest normalized	________	________

2.86 ◆

Consider a 16-bit floating-point representation based on the IEEE floating-point format, with one sign bit, seven exponent bits ($k = 7$), and eight fraction bits ($n = 8$). The exponent bias is $2^{7-1} - 1 = 63$.

Fill in the table that follows for each of the numbers given, with the following instructions for each column:

Hex: The four hexadecimal digits describing the encoded form.

M: The value of the significand. This should be a number of the form x or $\frac{x}{y}$, where x is an integer, and y is an integral power of 2. Examples include: 0, $\frac{67}{64}$, and $\frac{1}{256}$.

E: The integer value of the exponent.

V: The numeric value represented. Use the notation x or $x \times 2^z$, where x and z are integers.

As an example, to represent the number $\frac{7}{8}$, we would have $s = 0$, $M = \frac{7}{4}$, and $E = -1$. Our number would therefore have an exponent field of `0x3E` (decimal value $63 - 1 = 62$) and a significand field `0xC0` (binary 11000000_2), giving a hex representation `3EC0`.

You need not fill in entries marked "—".

Description	Hex	M	E	V
-0	____	____	____	—
Smallest value > 2	____	____	____	____
512	____	____	____	—
Largest denormalized	____	____	____	____
$-\infty$	____	—	—	—
Number with hex representation `3BB0`	—	____	____	____

2.87 ◆◆

Consider the following two 9-bit floating-point representations based on the IEEE floating-point format.

1. Format A
   - There is one sign bit.
   - There are $k = 5$ exponent bits. The exponent bias is 15.
   - There are $n = 3$ fraction bits.
2. Format B
   - There is one sign bit.
   - There are $k = 4$ exponent bits. The exponent bias is 7.
   - There are $n = 4$ fraction bits.

Below, you are given some bit patterns in Format A, and your task is to convert them to the closest value in Format B. If rounding is necessary, you should *round toward* $+\infty$. In addition, give the values of numbers given by the Format A and Format B bit patterns. Give these as whole numbers (e.g., 17) or as fractions (e.g., 17/64 or $17/2^6$).

Format A		Format B	
Bits	Value	Bits	Value
`1 01111 001`	$\frac{-9}{8}$	`1 0111 0010`	$\frac{-9}{8}$
`0 10110 011`	____	____	____
`1 00111 010`	____	____	____
`0 00000 111`	____	____	____
`1 11100 000`	____	____	____
`0 10111 100`	____	____	____

2.88 ◆

We are running programs on a machine where values of type `int` have a 32-bit two's-complement representation. Values of type `float` use the 32-bit IEEE format, and values of type `double` use the 64-bit IEEE format.