example, a constant that ends with U (or u) must have one of the types `unsigned int`, `unsigned long int`, or `unsigned long long int`. A decimal constant that ends with L (or `l`) must have one of the types `long int` or `long long int`. There's also a provision for a constant to have an extended integer type if it's too large to represent using one of the standard integer types.

Integer Overflow

When arithmetic operations are performed on integers, it's possible that the result will be too large to represent. For example, when an arithmetic operation is performed on two `int` values, the result must be able to be represented as an `int`. If the result can't be represented as an `int` (because it requires too many bits), we say that *overflow* has occurred.

The behavior when integer overflow occurs depends on whether the operands were signed or unsigned. When overflow occurs during an operation on *signed* integers, the program's behavior is undefined. Recall from Section 4.4 that the consequences of undefined behavior may vary. Most likely the result of the operation will simply be wrong, but the program could crash or exhibit other undesirable behavior.

When overflow occurs during an operation on *unsigned* integers, though, the result *is* defined: we get the correct answer modulo 2^n, where n is the number of bits used to store the result. For example, if we add 1 to the unsigned 16-bit number 65,535, the result is guaranteed to be 0.

Reading and Writing Integers

Suppose that a program isn't working because one of its `int` variables is overflowing. Our first thought is to change the type of the variable from `int` to `long int`. But we're not done yet; we need to see how the change will affect the rest of the program. In particular, we must check whether the variable is used in a call of `printf` or `scanf`. If so, the format string in the call will need to be changed, since the `%d` conversion works only for the `int` type.

Reading and writing unsigned, short, and long integers requires several new conversion specifiers:

Q&A

- When reading or writing an *unsigned* integer, use the letter `u`, `o`, or `x` instead of `d` in the conversion specification. If the `u` specifier is present, the number is read (or written) in decimal notation; `o` indicates octal notation, and `x` indicates hexadecimal notation.

```
unsigned int u;

scanf("%u", &u);    /* reads  u in base 10 */
printf("%u", u);    /* writes u in base 10 */
scanf("%o", &u);    /* reads  u in base  8 */
printf("%o", u);    /* writes u in base  8 */
```