```
scanf("%x", &u);    /* reads  u in base 16 */
printf("%x", u);    /* writes u in base 16 */
```

- When reading or writing a *short* integer, put the letter `h` in front of `d`, `o`, `u`, or `x`:

```
short s;

scanf("%hd", &s);
printf("%hd", s);
```

- When reading or writing a *long* integer, put the letter `l` ("ell," not "one") in front of `d`, `o`, `u`, or `x`:

```
long l;

scanf("%ld", &l);
printf("%ld", l);
```

C99

- When reading or writing a *long long* integer (C99 only), put the letters `ll` in front of `d`, `o`, `u`, or `x`:

```
long long ll;

scanf("%lld", &ll);
printf("%lld", ll);
```

PROGRAM

Summing a Series of Numbers (Revisited)

In Section 6.1, we wrote a program that sums a series of integers entered by the user. One problem with this program is that the sum (or one of the input numbers) might exceed the largest value allowed for an `int` variable. Here's what might happen if the program is run on a machine whose integers are 16 bits long:

```
This program sums a series of integers.
Enter integers (0 to terminate): 10000 20000 30000 0
The sum is: -5536
```

The sum was 60,000, which wouldn't fit in an `int` variable, so overflow occurred. When overflow occurs with signed numbers, the outcome is undefined. In this case, we got an apparently meaningless number. To improve the program, let's switch to `long` variables.

sum2.c

```
/* Sums a series of numbers (using long variables) */

#include <stdio.h>

int main(void)
{
  long n, sum = 0;

  printf("This program sums a series of integers.\n");
```