a suffix to n containing a combination of L and/or U characters. (See Section 7.1 for a discussion of how to use the L and U suffixes with integer constants.)

Section 27.2

3. (C99) In each of the following statements, assume that the variable i has the indicated original type. Using macros from the <inttypes.h> header, modify each statement so that it will work correctly if the type of i is changed to the indicated new type.

(a) `printf("%d", i);`	Original type: `int`	New type: `int8_t`
(b) `printf("%12.4d", i);`	Original type: `int`	New type: `int32_t`
(c) `printf("%-6o", i);`	Original type: `unsigned int`	New type: `uint16_t`
(d) `printf("%#x", i);`	Original type: `unsigned int`	New type: `uint64_t`

Section 27.5

4. (C99) Assume that the following variable declarations are in effect:

```
int i;
float f;
double d;
long double ld;
float complex fc;
double complex dc;
long double complex ldc;
```

Each of the following is an invocation of a macro in <tgmath.h>. Show what it will look like after preprocessing, when the macro has been replaced by a function from <math.h> or <complex.h>.

(a) `tan(i)`
(b) `fabs(f)`
(c) `asin(d)`
(d) `exp(ld)`
(e) `log(fc)`
(f) `acosh(dc)`
(g) `nexttoward(d, ld)`
(h) `remainder(f, i)`
(i) `copysign(d, ld)`
(j) `carg(i)`
(k) `cimag(f)`
(l) `conj(ldc)`

Programming Projects

1. (C99) Make the following modifications to the quadratic.c program of Section 27.4:

(a) Have the user enter the coefficients of the polynomial (the values of the variables a, b, and c).

(b) Have the program test the discriminant before displaying the values of the roots. If the discriminant is negative, have the program display the roots in the same way as before. If it's nonnegative, have the program display the roots as real numbers (without an imaginary part). For example, if the quadratic equation is $x^2 + x - 2 = 0$, the output of the program would be

```
root1 = 1
root2 = -2
```