```c
#include "complex.h"

/*
* Complex number input function returns standard scanning error code
* 1 => valid scan, 0 => error, negative EOF value => end of file
*/

int
scan_complex(complex_t *c) /* output - address of complex variable to fill */

{
    int status;

    status = scanf("%lf%lf", &c->real, &c->imag);
    if (status == 2)
        status = 1;
    else if (status != EOF)
        status = 0;

    return (status);
}

/*
* Complex output function displays value as (a + bi) or (a - bi),
* dropping a or b if they round to 0 unless both round to 0
*/

void
print_complex(complex_t c) /* input - complex number to display */

{
    double a, b;
    char sign;

    a = c.real;
    b = c.imag;

    printf("(");

    if (fabs(a) < .005 && fabs(b) < .005) {
            printf("%.2f", 0.0);
    } else if (fabs(b) < .005) {
            printf("%.2f", a);
    } else if (fabs(a) < .005) {
            printf("%.2fi", b);
    } else {
        if (b < 0)
            sign = '-';
        else
            sign = '+';
        printf("%.2f %c %.2fi", a, sign, fabs(b));
}

    printf(")");

}

/*
* Returns sum of complex values c1 and c2
*/

complex_t
add_complex(complex_t c1, complex_t c2) /* input - values to add */

{
    complex_t csum;

    csum.real = c1.real + c2.real;
    csum.imag = c1.imag + c2.imag;
    return (csum);
}
```

```c
/*
* Returns difference c1 -
*/

complex_t
subtract_complex(complex_t c1, complex_t c2) /* input parameters*/

{
    complex_t cdiff;
    cdiff.real = c1.real - c2.real;
    cdiff.imag = c1.imag - c2.imag;

    return (cdiff);
}

/* ** Stub **
* Returns product of complex values c1 and c2
*/

complex_t
multiply_complex(complex_t c1, complex_t c2) /* input parameters */

{
    printf("Function multiply_complex returning first argument\n");
    return (c1);
}

/* ** Stub **
* Returns quotient of complex values (c1 / c2)
*/

complex_t
divide_complex(complex_t c1, complex_t c2) /* input parameters */

{
    printf("Function divide_complex returning first argument\n");
    return (c1);
}

/*
* Returns absolute value of complex number c
*/

complex_t
abs_complex(complex_t c) /* input parameter */

{
    complex_t cabs;

    cabs.real = sqrt(c.real * c.real + c.imag * c.imag);
    cabs.imag = 0;

    return (cabs);
}
```