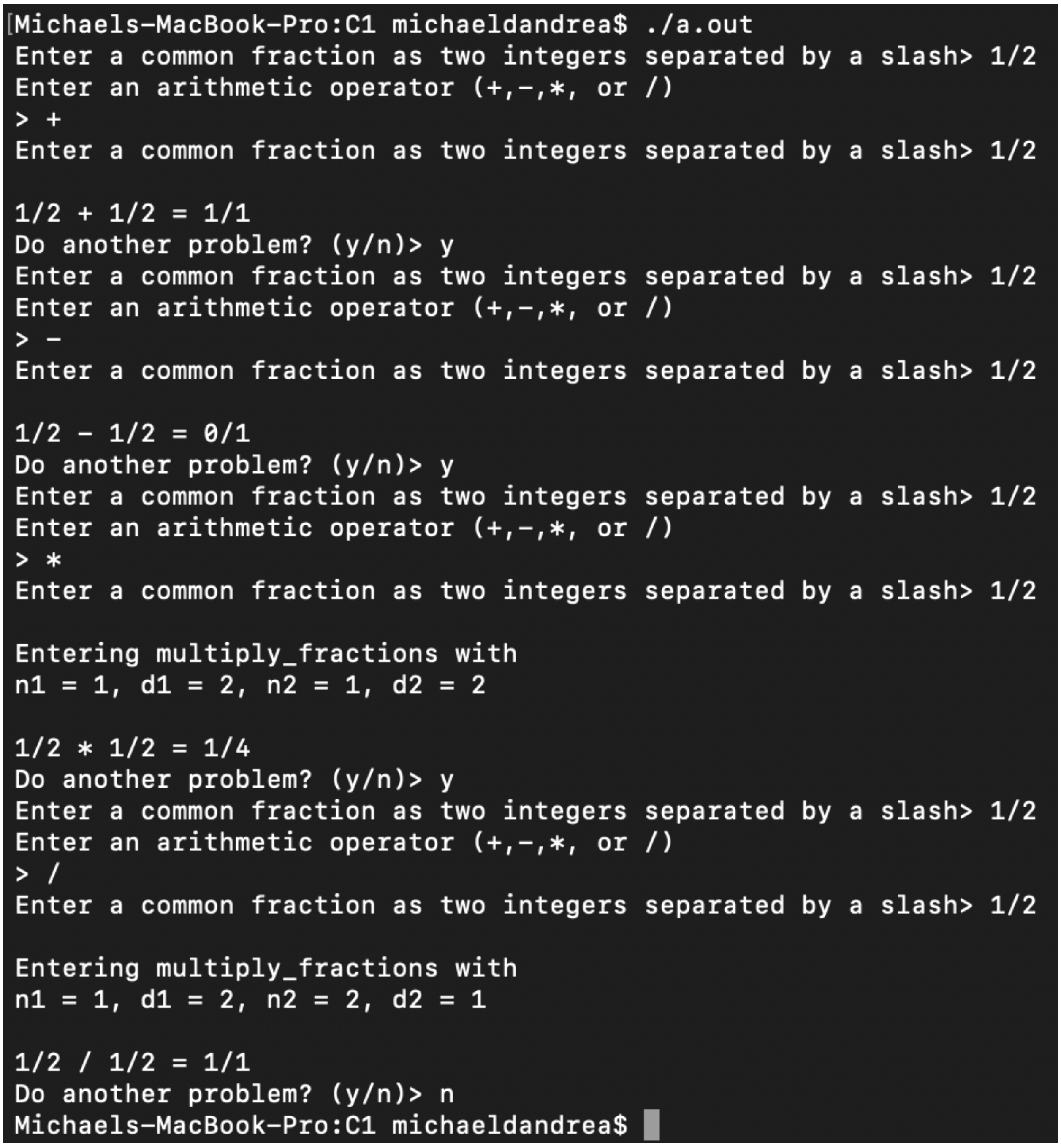
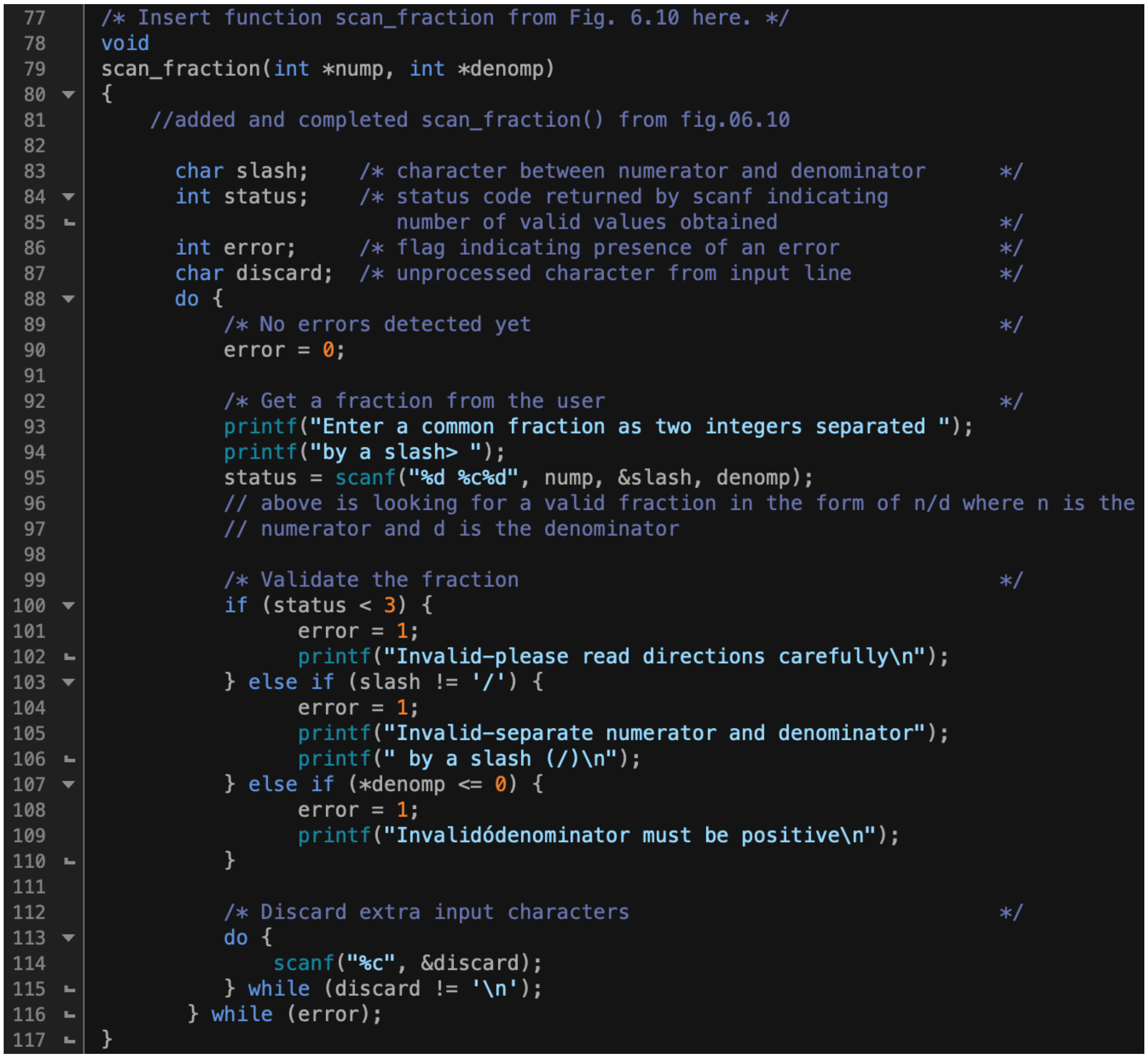
This program now asks for the first fraction, the operation to be performed, then the second fraction. It then does the operations to the two fractions and outputs the answer in reduced form.

Output of the program:



```
[Michaels-MacBook-Pro:C1 michaeldandrea$ ./a.out
Enter a common fraction as two integers separated by a slash> 1/2
Enter an arithmetic operator (+,-,*, or /)
> +
Enter a common fraction as two integers separated by a slash> 1/2

1/2 + 1/2 = 1/1
Do another problem? (y/n)> y
Enter a common fraction as two integers separated by a slash> 1/2
Enter an arithmetic operator (+,-,*, or /)
> -
Enter a common fraction as two integers separated by a slash> 1/2

1/2 - 1/2 = 0/1
Do another problem? (y/n)> y
Enter a common fraction as two integers separated by a slash> 1/2
Enter an arithmetic operator (+,-,*, or /)
> *
Enter a common fraction as two integers separated by a slash> 1/2

Entering multiply_fractions with
n1 = 1, d1 = 2, n2 = 1, d2 = 2

1/2 * 1/2 = 1/4
Do another problem? (y/n)> y
Enter a common fraction as two integers separated by a slash> 1/2
Enter an arithmetic operator (+,-,*, or /)
> /
Enter a common fraction as two integers separated by a slash> 1/2

Entering multiply_fractions with
n1 = 1, d1 = 2, n2 = 2, d2 = 1

1/2 / 1/2 = 1/1
Do another problem? (y/n)> n
Michaels-MacBook-Pro:C1 michaeldandrea$ 
```

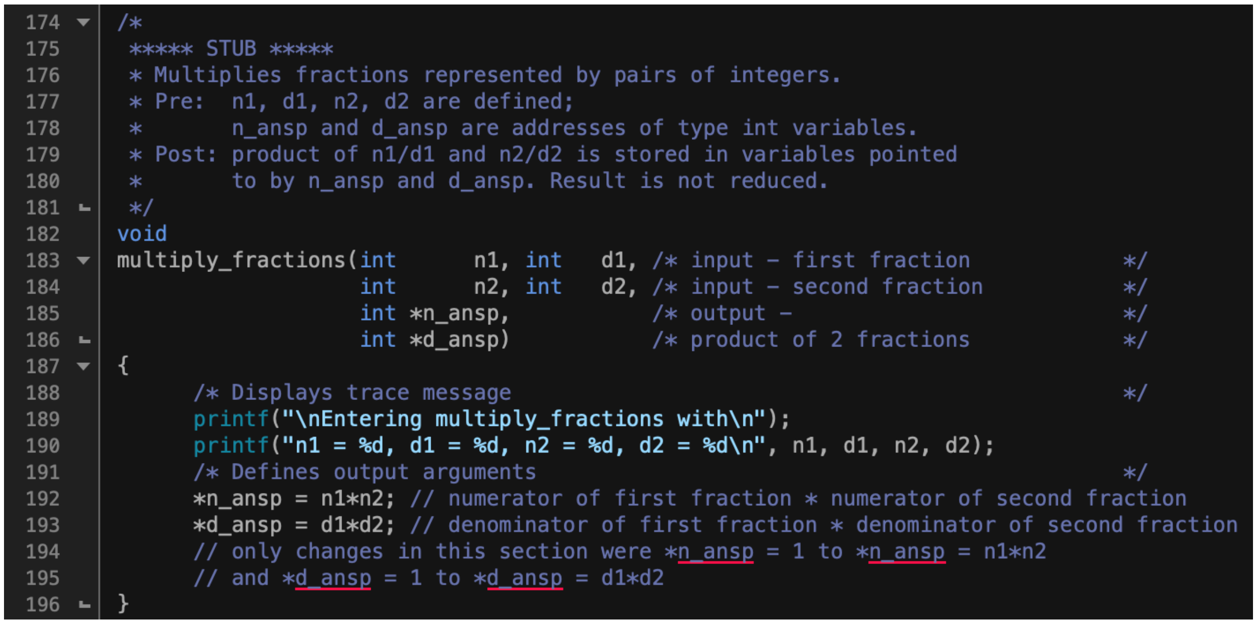
To complete this program, I had to implement the scan_fraction() function from fig.06.10. It is implemented as follows:



```
 77 /* Insert function scan_fraction from Fig. 6.10 here. */
 78 void
 79 scan_fraction(int *nump, int *denomp)
 80 {
 81     //added and completed scan_fraction() from fig.06.10
 82
 83       char slash;    /* character between numerator and denominator      */
 84       int status;    /* status code returned by scanf indicating
 85                         number of valid values obtained                  */
 86       int error;     /* flag indicating presence of an error             */
 87       char discard;  /* unprocessed character from input line            */
 88       do {
 89           /* No errors detected yet                                      */
 90           error = 0;
 91
 92           /* Get a fraction from the user                                */
 93           printf("Enter a common fraction as two integers separated ");
 94           printf("by a slash> ");
 95           status = scanf("%d %c%d", nump, &slash, denomp);
 96           // above is looking for a valid fraction in the form of n/d where n is the
 97           // numerator and d is the denominator
 98
 99           /* Validate the fraction                                       */
100           if (status < 3) {
101                 error = 1;
102                 printf("Invalid-please read directions carefully\n");
103           } else if (slash != '/') {
104                 error = 1;
105                 printf("Invalid-separate numerator and denominator");
106                 printf(" by a slash (/)\n");
107           } else if (*denomp <= 0) {
108                 error = 1;
109                 printf("Invalidódenominator must be positive\n");
110           }
111
112           /* Discard extra input characters                              */
113           do {
114               scanf("%c", &discard);
115           } while (discard != '\n');
116        } while (error);
117 }
```

The outline of the function was copied from fig.06.10, and I added the “nump, &slash, denomp” for reasons stated in the comment on lines 96 and 97.

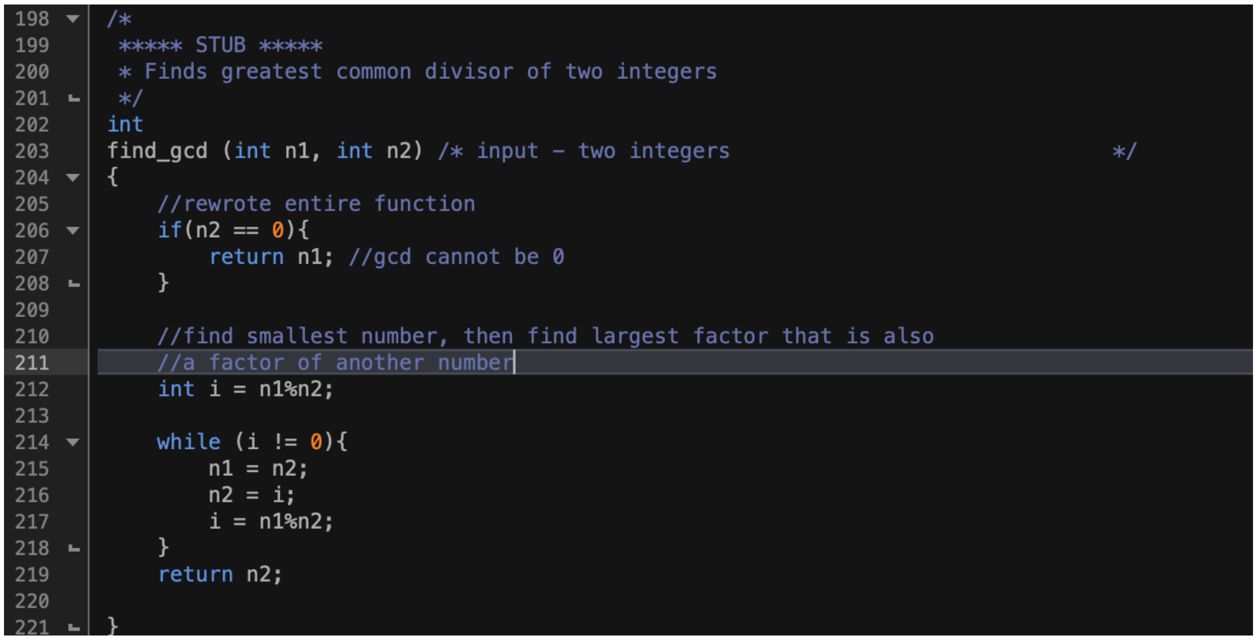
I also edited the multiply_fractions() function as follows:



```c
/*
 ***** STUB *****
 * Multiplies fractions represented by pairs of integers.
 * Pre:  n1, d1, n2, d2 are defined;
 *       n_ansp and d_ansp are addresses of type int variables.
 * Post: product of n1/d1 and n2/d2 is stored in variables pointed
 *       to by n_ansp and d_ansp. Result is not reduced.
 */
void
multiply_fractions(int      n1, int   d1, /* input - first fraction         */
                   int      n2, int   d2, /* input - second fraction        */
                   int *n_ansp,           /* output -                       */
                   int *d_ansp)           /* product of 2 fractions         */
{
      /* Displays trace message                                            */
      printf("\nEntering multiply_fractions with\n");
      printf("n1 = %d, d1 = %d, n2 = %d, d2 = %d\n", n1, d1, n2, d2);
      /* Defines output arguments                                          */
      *n_ansp = n1*n2; // numerator of first fraction * numerator of second fraction
      *d_ansp = d1*d2; // denominator of first fraction * denominator of second fraction
      // only changes in this section were *n_ansp = 1 to *n_ansp = n1*n2
      // and *d_ansp = 1 to *d_ansp = d1*d2
}
```

This change was to make the program multiply the fractions, rather than having the numerator and denominator always return as 1.

The final change was to the find_gcd() function as follows:



```c
/*
 ***** STUB *****
 * Finds greatest common divisor of two integers
 */
int
find_gcd (int n1, int n2) /* input - two integers                          */
{
    //rewrote entire function
    if(n2 == 0){
        return n1; //gcd cannot be 0
    }

    //find smallest number, then find largest factor that is also
    //a factor of another number
    int i = n1%n2;

    while (i != 0){
        n1 = n2;
        n2 = i;
        i = n1%n2;
    }
    return n2;

}
```

This entire function was re-written because I could not understand how the authors of the book wanted us to finish the function. This function finds the largest factor of the number that is also a factor of the other number.