```c
// Filename Fread_Program6.c
// Written by Justin Fread
// Written on 10/24/18

#include <stdio.h>
#include <stdlib.h>

// Function Prototypes
void scanFraction(int *ptrNum, int *ptrDenom);
char getOperator(void);
void addFractions(int n1, int d1, int n2, int d2, int *ptr_nAns, int *ptr_dAns);
void multiplyFractions(int n1, int d1, int n2, int d2, int *ptr_nAns, int *ptr_dAns);
int  find_gcd(int n1, int n2);
void reduceFraction(int *ptrNum, int *ptrDenom);
void printFraction(int num, int denom);

int main(void) {

	int n1;
	int d1;
	int n2;
	int d2;
	int nAnswer;
	int dAnswer;
	char option;
	char again;

	again = 'y';

	// While the user wants to continue, gets and solves arithmitic problems
	// with common fractions
	while (again == 'Y' || again == 'y') {

		// Gets a fraction problem
		scanFraction(&n1, &d1);
		option = getOperator();
		scanFraction(&n2, &d2);

		// Computes the result
		switch (option) {
		case '+':
			addFractions(n1, d1, n2, d2, &nAnswer, &dAnswer);
			break;
		case '-':
			addFractions(n1, d1, -n2, d2, &nAnswer, &dAnswer);
			break;
		case '*':
			multiplyFractions(n1, d1, n2, d2, &nAnswer, &dAnswer);
			break;
		case '/':
			multiplyFractions(n1, d1, n2, d2, &nAnswer, &dAnswer);
			break;
		}
		reduceFraction(&nAnswer, &dAnswer);

		// Display problem and results
		printf("\n");
		printFraction(n1, d1);
		printf(" %c ", option);
		printFraction(n2, d2);
		printf(" = ");
		printFraction(nAnswer, dAnswer);
		printf("\n");

		// Promt user to enter another problem or quit
		printf("\nDo another problem? (y/n)--> ");
		scanf_s("%c", &again);

	}

	system("pause");
}

// Gets and returns a valid fraction as its result
void scanFraction(int *ptrNum, int *ptrDenom) {

	char slash;
	int status;
	int error;
	char discard;

	do {
		error = 0;

		// Get a fraction from the user
		printf("Enter a common fraction as two integers seperated ");
		printf("by a slash--> ");
		status = scanf_s("%d", ptrNum);
		status += scanf_s("%c", &slash);
		status += scanf_s("%d", ptrDenom);

		// Validate the fraction
		if (status < 3) {
			error = 1;
			printf("Invalid-please read directions carefully\n");
		}
		else if (slash != '/') {
			error = 1;
			printf("Invalid-seperate numerator and denominator");
			printf(" with a slash (/)\n");
		}
		else if (ptrDenom <= 0) {
			error = 1;

			printf("Invalid-denominator must be positive\n");
		}
		// Discard extra input characters
		do {
			scanf_s("%c", &discard);
		} while (discard != '\n');

	} while (error);

}

// Gets and returns a valid arithmitic operator, skips over newline charactors
// and permits reentry of operator incase of error
char getOperator(void) {

	char option;

	printf("Enter an arithmitic operator (+, -, *, or /)\n-->");

	for (scanf_s("%c", &option);
		option != '+' && option != '-' &&
		option != '*' && option != '/';
		scanf_s("%c", &option)) {

		if (option != '\n') {
			printf("%c invalid, reenter operator (+, -, *, or /)\n-->", option);
		}

	}

	return option;
}

// Adds fractions represented by pairs of integers
void addFractions(int n1, int d1, int n2, int d2, int *ptr_nAns, int *ptr_dAns) {

	int denom;
	int numer;
	int signFactor;

	// finds a common denominator
	denom = d1 * d2;

	// computes numerator
	numer = n1 * d2 + n2 * d1;

	// Adjust sign (at most, numerator should be negative)
	if (numer * denom >= 0) {
		signFactor = 1;
	}
	else {
		signFactor = -1;
	}

	numer = signFactor * abs(numer);
	denom = abs(denom);

	// Return results
	*ptr_nAns = numer;
	*ptr_dAns = denom;

}

// Multiplies fractions represented by pairs of integers
void multiplyFractions(int n1, int d1, int n2, int d2, int *ptr_nAns, int *ptr_dAns) {

	// Displays trace message
	printf("\nEntering multiplyFractions with\n");
	printf("n1 = %d, n2 = %d, d1 = %d, d2 = %d", n1, n2, d1, d2);

	// Defines output arguments
	*ptr_nAns = 1;
	*ptr_dAns = 1;

}

// Finds greatest common devisor of two integers
int find_gcd(int n1, int n2) {

	int gcd;
	int q;
	int p;
	int r;

	// detemine absolute values of n1 and n2 and place them in q and p respectievly
	q = abs(n1);
	p = abs(n2);
	r = q % p;

	// calculate gcd
	while (r != 0) {
		q = p;
		p = r;
		r = q % p;
	}

	gcd = p;

	return gcd;

}

// Reduces a fraction by dividing its numerator and denominator by its

// greatest common devisor
void reduceFraction(int *ptrNum, int *ptrDenom) {

	int gcd;

	gcd = find_gcd(*ptrNum, *ptrDenom);

	*ptrNum = *ptrNum / gcd;
	*ptrDenom = *ptrDenom / gcd;

}

// Displays a pair of integers as a fraction
void printFraction(int num, int denom) {

	printf("%d/%d", num, denom);

}
```