```
/*
 * Program for multiplying and dividing complex numbers.
 * Author: Jeet Chakrabarty
 */


#include <stdio.h>  //Includes inputs and outputs
#include <stdlib.h> //Includes structs

//Defines a structure of type complex_t
struct complex_t
	{
		double real;	//Sets attribute denoting real portion of complex number
		double imaginary;   //Sets attribute denoting imaginary portion of complex number
	};

//Function to multiply two complex numbers
struct complex_t multiply(struct complex_t one, struct complex_t two)
{
   //Defines a variable of type complex_t to be returned
	struct complex_t ret;

	//Sets the real portion of the complex number according to formula given
	ret.real= one.real*two.real-one.imaginary*two.imaginary;

	//Sets the imaginary portion of the complexnumber according to formula given
	ret.imaginary = two.real*one.imaginary+one.real*two.imaginary;

	return ret; //Returns value
```

```
}

//Function to divide two complex numbers
struct complex_t *division(struct complex_t *one, struct complex_t *two)
{
   //Defines a variable to be returned while allocating memory for pointer to point to
        struct complex_t *ret = malloc(sizeof (struct complex_t));

        //Sets variable to store square portion of formula
        double square = (two->real)*(two->real) + (two->imaginary)*(two->imaginary);

        //Executes if formula valid (square portion not equal to 0)
        if (square != 0)
        {
           //Sets values pointer points to according to formulae
                ret->real = (one->real*two->real + one->imaginary*two->imaginary)/square;
                ret->imaginary = (two->real*one->imaginary-one->real*two->imaginary)/square;
        }
```

```
/*
 * Program for multiplying and dividing complex numbers.
 * Author: Jeet Chakrabarty
 */


#include <stdio.h>  //Includes inputs and outputs
#include <stdlib.h> //Includes structs

//Defines a structure of type complex_t
typedef struct
        {
                double real;       //Sets attribute denoting real portion of complex number
                double imaginary;   //Sets attribute denoting imaginary portion of complex number
        }complex_t;

//Function to multiply two complex numbers
complex_t multiply(complex_t one, complex_t two)
{
   //Defines a variable of type complex_t to be returned
        complex_t ret;

        //Sets the real portion of the complex number according to formula given
        ret.real= one.real*two.real-one.imaginary*two.imaginary;

        //Sets the imaginary portion of the complexnumber according to formula given
        ret.imaginary = two.real*one.imaginary+one.real*two.imaginary;

        return ret; //Returns value
```

```
}

//Function to divide two complex numbers
complex_t *division(complex_t *one, complex_t *two)
{
   //Defines a variable to be returned while allocating memory for pointer to point to
        complex_t *ret = malloc(sizeof (complex_t));

        //Sets variable to store square portion of formula
        double square = (two->real)*(two->real) + (two->imaginary)*(two->imaginary);

        //Executes if formula valid (square portion not equal to 0)
        if (square != 0)
        {
           //Sets values pointer points to according to formulae
                ret->real = (one->real*two->real + one->imaginary*two->imaginary)/square;
                ret->imaginary = (two->real*one->imaginary-one->real*two->imaginary)/square;
        }

        //Exits program if division by 0 (square portion is equal to 0)
        else
        {
           //Prints error message
                printf("Sorry, there's been an error.\n");
      //Exits program
                exit(0);
        }

        return ret; //Returns value to be returned
```

```
};

//Main part of program used to test functions
int main (void){
   //Declares variables to store inputs, product, and pointer for quotient
         complex_t a, b, c, *d;

   //Prompts user for inputs and stores as appropriate
         printf("Please enter the real portion of the first number\n");
         scanf("%lf", &a.real);
         printf("Please enter the imaginary portion of the first number\n");
         scanf("%lf", &a.imaginary);
         printf("Please enter the real portion of the second number\n");
         scanf("%lf", &b.real);
         printf("Please enter the imaginary portion of the second number\n");
         scanf("%lf", &b.imaginary);

         c = multiply(a, b);     //Stores product in c
         d = division(&a, &b);   //Stores quoteient in d pointer

         //Prints out results of multiplication and division
         printf("The product is equal to: \t %lf + i(%lf)\n", c.real, c.imaginary);
         printf("The quotient is equal to: \t %lf + i(%lf)\n", d->real, d->imaginary);
}
```

```
jharvard@appliance (~/Downloads): ./a.out
Please enter the real portion of the first number
2
Please enter the imaginary portion of the first number
3
Please enter the real portion of the second number
4
Please enter the imaginary portion of the second number
5
The product is equal to: -7.000000 + i(22.000000)
The quotient is equal to: 0.560976 + i(0.048780)
jharvard@appliance (~/Downloads): ./a.out
Please enter the real portion of the first number
2
Please enter the imaginary portion of the first number
0
Please enter the real portion of the second number
3
Please enter the imaginary portion of the second number
0
The product is equal to: 6.000000 + i(0.000000)
The quotient is equal to: 0.666667 + i(0.000000)
jharvard@appliance (~/Downloads): ./a.out
Please enter the real portion of the first number
0
Please enter the imaginary portion of the first number
1
Please enter the real portion of the second number
0
```

Please enter the imaginary portion of the second number1

The product is equal to: -1.000000 + i(0.000000)

The quotient is equal to: 1.000000 + i(0.000000)

jharvard@appliance (~/Downloads): ./a.out

Please enter the real portion of the first number

8.2345

Please enter the imaginary portion of the first number

0

Please enter the real portion of the second number

0

Please enter the imaginary portion of the second number

1

The product is equal to: 0.000000 + i(8.234500)

The quotient is equal to: 0.000000 + i(-8.234500)

jharvard@appliance (~/Downloads): ./a.out

Please enter the real portion of the first number

0

Please enter the imaginary portion of the first number

4

Please enter the real portion of the second number

234.324

Please enter the imaginary portion of the second number

0

The product is equal to: 0.000000 + i(937.296000)

The quotient is equal to: 0.000000 + i(0.017070)

jharvard@appliance (~/Downloads): ./a.out

Please enter the real portion of the first number

0

Please enter the imaginary portion of the first number

0Please enter the real portion of the second number

234.43

Please enter the imaginary portion of the second number

23.9

The product is equal to: 0.000000 + i(0.000000)

The quotient is equal to: 0.000000 + i(0.000000)

jharvard@appliance (~/Downloads): ./a.out

Please enter the real portion of the first number

324.76756

Please enter the imaginary portion of the first number

54.54673

Please enter the real portion of the second number

0

Please enter the imaginary portion of the second number

0

Sorry, there's been an error.