Objective:

This exercise will work with functions:

- Function definitions:
  - Value arguments
  - Reference arguments
  - Return values
- Function calls
  - Passing value arguments
  - Passing reference arguments

Description:

1. The user will enter three numbers (integers):
   - Two numbers to use in a math operation — `int num1, num2`
   - A number that indicates which math operation to perform. — `int operation`
   - A variable will be declared to hold the result of the math operation — `int result`
     i. This will get the return value from the function
   - In the case of division, a float result must be expected. A float variable will be defined — `float div`
     i. If the operation selected is DIVIDE, the function will return the result to the address of this variable.

Calling a Function:

2. These values and the address of the float variable will be passed to a function:

```
result = math_op(num1,num2,operation, &div);
```

Define a Function:

3. A function (ie **`math_op()`**) will be defined to receive the three integer values and the address of div

```
int math_op(int a, int b, int c, float *d)
```

Note: The function allusion should be placed at the top of the source file.

```
int math_op(int, int, int, float *)
```

- The function will perform this logic:
  i. The function will have a variable to hold the answer of the math — `int answer`
  ii. Determine which operation is to be performed. — `switch (c)`
  - If it is Add, Subtract, or Multiply, the return value will be set accordingly. — `case 1: (or 2, 3)`
    a. `answer = a+b //(or: a-b, a*b)`
  - If the operation is Divide — `case 4:`
    a. if the 2nd number is 0 — `if (b ==0)`
       i. set the return value to -98 and do not divide — `answer = -98.`
    b. if the 2nd number is not 0: — `else`
       i. set the return value to -97 — `answer = -97.`
       ii. divide a/b
       iii. have the result placed in the float address argument. — `*d = (float)a / b.`
  - If the operation is not 1,2,3, or 4
    a. Set the return value as -99 to indicate an invalid operation number was entered. — `answer = -99.`

Return Value:

4. The value of answer will be returned by the function: — `return answer.`

Process the Returned Value:

5. When the function is completed the main() function will check the return values for error codes.
   - If a -99 is returned, the error was an invalid operation. Print a message indicating the error.
   - If a -98 is returned, the error was a divide by zero. Print a message indicating the error.
   - If a -97 is returned, the calculation result will be in the float variable div. Print the result (`div`) using %.2f
   - Otherwise, the return value is the proper integer calculation result. Print the result (`result`) using %d.

Write comments:

- Your final code should include comments:
  i. A heading comment for the source file (Name, Purpose, Author, Date)
  ii. A heading for each function (main () and math_op()
    - Document, arguments and return values
  iii. Logic comments that indicate the logic of the actions
    - These are not to explain the statements, but to give an idea of the purpose of the action.

When completed

- Create a video of your program running to show the results generated.
  - The video does not have to explain the **code line by line**,
  - It would be good to **mention**:
    - the statement calling the function (Note the arguments passed by value and by reference)
    - the function definition (Note the variables received as values and as a pointer)
    - the function logic.
      a. Determining the operation to perform
      b. Using the float address for the division result.
      c. Setting the error codes.
    - handling the error codes returned by the function.

- **Upload** your c source file