1a. ip = &i;
    ipp = &ip;

1b. ip is a pointer and I'm setting the address it points to, to integer i's address with the reference operator.
    ipp is a pointer to a pointer, so I'm setting it's pointed address to ip's address. Then when I dereference ipp to **ipp, it
    follows the addresses down to integer i's, and then dereferences to get the value.

2a. g++ cast.cpp -fpermissive -o cast.exe

2b. Without cast: 51
      Cast 1: 102
      Cast 2: 408

2c. i = 0
    dp = 2000
    dbp = 2000
    cp = 1000
    cbp = 1000

2d. i = 0
    dp = 2408
    dbp = 2000
    cp = 1051
    cbp = 1000

2e. Line 20 prints 51 because the dp points to address 2408 and dbp points to address 2000, but since they are 8bytes per double, we divide 408/8 = 51
    Line 21 prints 102 because an int is half the size of a double. Therefore each time the pointer moves, it moves 4 bytes at a time instead of 8, so the gap is twice as big.
    Line 22 prints 408 because the program literally interprets the difference between memory addresses since we arent looking at blocks of bytes.

3a. the program should print rval: 33333 zero: 333

33 because the functions takes a parameter passed by reference, which allows the variable passed in to have its value manipulated, since each return calls another function that also passes by reference, the variable will continue to change until it returns the final result. func3 sets the original variable passed through to 33333 and rval is set equal to the return value of func1.

3b. If we remove the & in the func3 parameter then zero should become 22222

3c. I don't think there is a way to get rval = 11111 because theres no way to just return after the func1 by changing one character.

4. Compiling code takes the src code and interprets it to create an object file with machine code. Linkers take the include libraries and help the compiler understand what undeclared things in the cpp files are. Then the linker strings together each object file to create an executable file

4i. If you compiled MyClass.cpp it would go through the includes with the linker and then interpret the cpp file with the compiler and then link together each object file.

4ii. Since this is a single file, the Object would have to have everything about it above the main, so the linker would not have to be called if there are no include statements and there are no object files to link together.

5a. If its global then it will be static.

5b. It will be placed on the stack if its a local variable.

5c. It cant be placed on the heap because we can't use new.

6a. The reason getFloatArrayTwo doesnt work is bec
ause in the for loop your treating the float point
er as an array, it should be adding the size of ea
ch float to the pointer to put the next float 8 by
tes away from the previously pointed to memory add
ress location.

6b. Put a delete statement after the function showF
loatArray()