18) 1. Preprocessing. The preprocessor takes the program.c source code and produces an equivalent .c source code, performing operations such as removing comments.

2. **Compiling**. The compiler turns the preprocessed code into assembly code for the specific processor.

3. **Assembling**. The assembler converts the assembly instructions into processor-dependent machine-level binary object code.

4. **Linking**. The linker takes one or more object code files and produces a single executable file.

19) An int which is 0 if the program is successful and something else if it is not

21) 1) 140
2) 4
3) 24

22) a) 0
b)0 .667
c) 0.000
d)3
e)3
f)3.000

27) First, I will check each individual function to see if I get what I expect. If I do then I will look at the data types. Depending on the error this could cause overflow or problems printing. I will also look at the variables and loops and make sure they are working by putting prints to see if they give the values I expect when I run the program.

28) done

30) a) 3
b) 4
c) 2
d) 6
e) error/unknown
f) error/unknown
g) 2

31) i = 3*1 + 2 + 0
i=5
This is the case because the Booleans return 0 or 1 for false and true

32) a) 0xF2
b) 0x01

c)0x0F
d)0x0E
e)0x01
f)0x68
g) 0x00

34)asci.c

35)bubb.c