ME 333
Winter Break Assignment

1. HelloWorld.c attached

2. A pointer variable contains the memory address of another variable as opposed to that variable's value. A non-pointer variable just contains the value. Pointers can be used to access the contents of the address it stores.

3. Interpreted code is converted to machine code and executed all while the program is running, while compiled code converts the code to machine code and stores the converted code ahead of time. Then, later, that compiled code can be executed which makes it faster to run because it doesn't need to do the conversion.

4.

a) Hexadecimal (base 16): 0x1E
   Eight-digit binary (base 2): 0b11110
   Three-digit decimal (base 10): 30

b) Hexadecimal: 0x32
   Eight-digit binary: 110010
   Three-digit decimal: 50

c) Hexadecimal: 0xFE
   Eight-digit binary: 11111110
   Three-digit decimal: 254

d) Hexadecimal: 0xC4
   Eight-digit binary: 11000100
   Three-digit decimal: 196

6. Assuming that each byte of memory can be addressed by a 16-bit address, and every 16-bit address has a corresponding byte in memory, I have 2^{16} total bits of memory.

7.

a) ch = 'k' → 107
b) ch = '5' → 53
c) ch = '=' → 61
d) ch = '?' → 63

8.

a) unsigned char: 0 to 255
b) short: -2^{15} to $2^{15} - 1$
c) double: -2^{63} to $2^{63} - 1$

10. Unsigned means only using nonnegative integers, whereas signed can be both negative and nonnegative.
Ex. unsigned short has values ranging from 0 to 2^{16} - 1, whereas signed short has values ranging from -2^{15} to 2^{15} - 1

11.

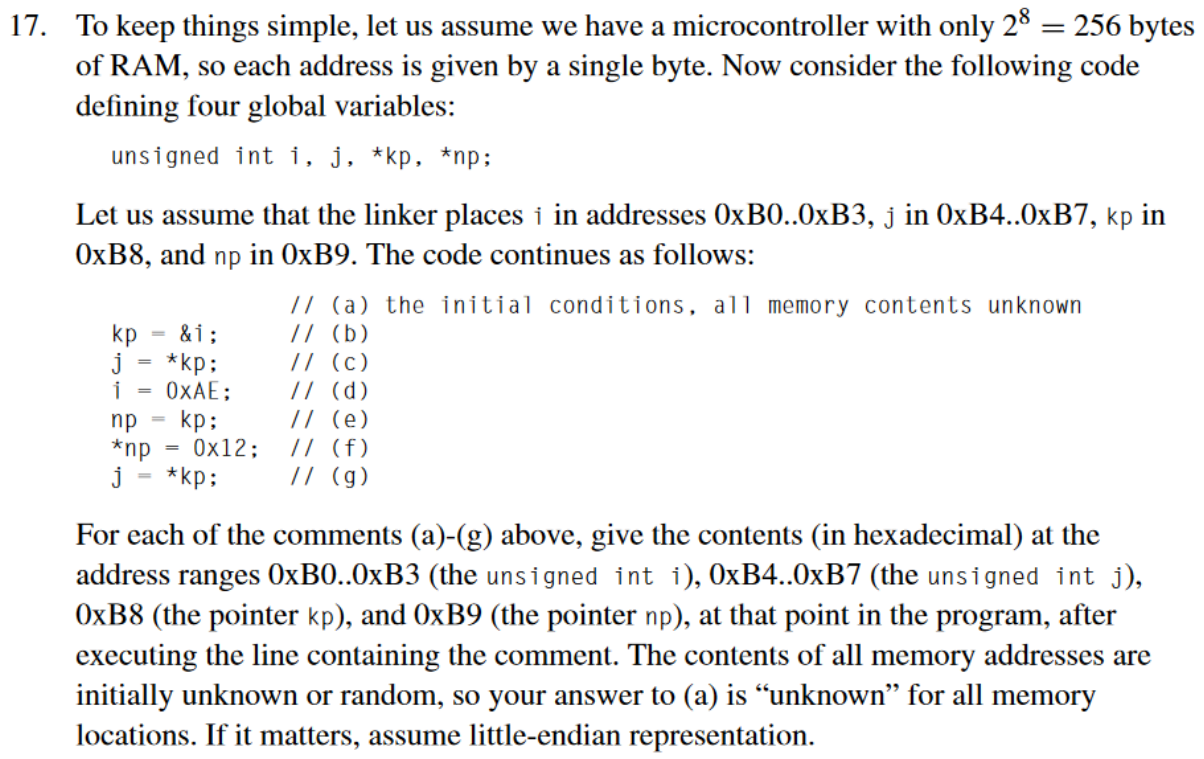
a) chars vs ints for integer math
Chars are better in that they have lower memory usage and a potential faster computation time, while ints are better in terms of the possible values that can be represented.

b) floats vs doubles for floating point math
Using floats is beneficial because they use less memory because it is a smaller data type and have potentially faster computations, while using doubles is beneficial because they have greater precision in the representation compared to floats.

c) chars vs floats for integer math
Using chars is preferable because integer math is faster, and integer data types are able to represent larger integers for the same amount of bytes. However, using a float could be better if you need a larger range of integer values, since floats have more memory allocation than chars.

16. Assuming pointers occupy eight bytes, they are the most similar to the float data type because it also occupies eight bytes.

17.



17. To keep things simple, let us assume we have a microcontroller with only $2^8 = 256$ bytes of RAM, so each address is given by a single byte. Now consider the following code defining four global variables:

```
unsigned int i, j, *kp, *np;
```

Let us assume that the linker places `i` in addresses 0xB0..0xB3, `j` in 0xB4..0xB7, `kp` in 0xB8, and `np` in 0xB9. The code continues as follows:

```
              // (a) the initial conditions, all memory contents unknown
kp = &i;      // (b)
j = *kp;      // (c)
i = 0xAE;     // (d)
np = kp;      // (e)
*np = 0x12;   // (f)
j = *kp;      // (g)
```

For each of the comments (a)-(g) above, give the contents (in hexadecimal) at the address ranges 0xB0..0xB3 (the `unsigned int i`), 0xB4..0xB7 (the `unsigned int j`), 0xB8 (the pointer `kp`), and 0xB9 (the pointer `np`), at that point in the program, after executing the line containing the comment. The contents of all memory addresses are initially unknown or random, so your answer to (a) is "unknown" for all memory locations. If it matters, assume little-endian representation.

	Contents at 0xB0…0xB3 (i)	Contents at 0xB4…0xB7 (j)	Contents at 0xB8 (kp)	Contents at 0xB9 (np)
a)	unknown	unknown	unknown	unknown
b)	unknown	unknown	0xB0	unknown
c)	unknown	unknown	0xB0	unknown
d)	0xAE	unknown	0xB0	unknown
e)	0xAE	unknown	0xB0	0xB0
f)	0x12	unknown	0xB0	0xB0
g)	0x12	0x12	0xB0	0xB0