Solutions 7

Question 1:

a) *(uint16_t*)0x2000001C = (*(uint16_t*)0x2000001C | 0x0600) & 0xFFC7; 3 marks

b)
```
        LDR R0, ADDR
        LDRH R1, [R0]
        LDR R2, MASK_IN
        ORRS R1, R1, R2
        LDR R2, MASK_OUT
        ANDS R1, R1, R2
        STRH R1, [R0]
        .align
ADDR: .word 0x2000001C
MASK_IN: .word 0x0600
MASK_OUT: .word 0xFFC8 @if we load a word from 0x2000001C then this should be 0xFFFFFFC8 otherwise we must STRH to the address - Ross
```

3 Marks

c) The assignment operator. Equals. =. Reason: for a pin to be an output, the two bits associated with that pin must be 01. Hence, simply setting some bits high may not be sufficient, the other bits must be forced to 0 as well. Put differently: it's necessary to define specific values for all bits. 2 Marks

d) When you use (ie: call) a function, the compiler wants to check that you're using it correctly. A prototype defines the interface to the function so the compiler be able to check that it's being called correctly. 2 Marks

Question 2:

a) A variable defined inside a function. (more correctly: without the static qualified. But they hadn't been taught that yet.) 1 Marks

b) On the stack. The stack pointer is decremented by the necessary amount. 2 Marks

c) The lifespan of the variable is the duration of the function call. When the function is called, the space on the stack is allocated for the variable. When the function returns the space is deallocated. The advantage of this is that, by not having a fixed memory address, we get more efficient memory usage as variable only occupy space in memory when they are being used. 3 Marks

d) The data type of the variable. 1 Marks

e)
```
if ((foo & 0b10000000000) != 0) {
        // some stuff
}
```

in hex: 0x400   with bit shift: (1 << 10)   decimal: 1024 2 Marks