2) The most lucid way I think for me to explain what the mystery function does is that it goes to a at its address at b (that is b + ptr, which is 2 bytes over from the position of a(or any number of bytes over as long as its a number with in bounds, which the if statement in the function will change an out of bounds b argument to 0 and in that case the function will just replace the byte the pointer specifically points to which will be different on little endian machines vs big endian and replaces what is in that byte. The better name might be "put\_c\_in\_a\_at\_position\_b".

3) For this problem the function mystery get an integer n and then completely swaps the HEX values. In essence it changes the representation of a small endian to a large endian value on the intel I used to work on it. For example this function could be used to change int data when an intel were sending data to a ppc.

In the example input/output shown 1234 becomes -771489792 on the same machine. when looking at the HEX values it becomes obvious though. 1234(0x000004d2) becomes -771489792(0xd2040000). every 2 byte hex block is switched around. Specifically the statement in line 25 accomplishes this by shifting the 32-bits around in a mirrored fashion.