Tutorial 9

Question 1: (9)

a) Assume we declare a struct as follows. Write a line of C code which will define a variable called `foobar` of type: pointer to SomeStruct. The variable `foobar` should be initialised to point to address 0x2000 0400. (2)

```
struct SomeStruct {
      uint32_t  a;
      uint16_t  b[8];
      uint8_t  *c;
      uint16_t  d[2];
};
```

b) Following on from the above, which address(es) will be modified by the following line of code? (2)

```
*(foobar).d[1] = 0;
```

c) Re-write the above using the member indirect access operator. (1)

d) Explain how a #defined works, making reference to how it's used in the header file provided by ST. (2)

e) Describe what happens when we use a #include compiler directive in a source file. (2)

Question 2: (14)

a) What is the difference between the VMA and LMA of a section? (3)

b) Making reference to the above, explain how statically allocated variables are initialised. (2)

c) Discuss the efficiency introduced by splitting the statically allocated variables into a .data section and a .bss section. (2)

d) The keyword 'static' can have two meanings depending on how it's used: static allocation or static visibility / linkage. Explain what the two different meaning are, mentioning both variables and functions, and explain why static visibility should be implemented when possible. (5)

e) In class I presented a viewpoint which said that global variables should be avoided whenever possible. Discuss one instance where it is necessary to use global variables, making reference to why we have no choice in this instance. (2)