Write the appropriate instruction to allocate stack space for the following local variables and any padding. All of these local variables must be allocated on the runtime stack in the order discussed in class.

```
int a; // 1st local var
unsigned short b; // 2nd local var
char c; // 3rd local var
short d; // 4th local var
char e; // 5th local var
int f; // 6th local var
```

```
push {fp, lr}   @ saved registers
add fp, sp, 4   @ set fp to point to the base of the saved regs
_____ _____, _____, _____ @ allocate space for the above local vars
```

Write the appropriate unoptimized ARM assembly instructions using the above local variables. Treat every statement separately and independently. Do not assume any value in any register from a previous statement is available in a later statement.

```
a = c;  _________________________________

b = 0xDEAD; _________________________________

e = 'A';  _________________________________
```

Use magic numbers and direct offsets from the frame pointer vs. assembler constants or other calculations.

```
sp needs to be kept on an evenly divisible by _____ byte boundary in most architectures including ARM.
```

Affirm your adherence to the principle of Academic Integrity by writing the following statement:

"I Excel with Integrity"

Score:
Write the appropriate ARM instruction to allocate stack space for the following local variable declarations in the order discussed in class.

```assembly
short a[5]; // 1st local var on stack
short *ptr; // 2nd local var on stack
```

```
push {fp, lr} @ saved registers
add fp, sp, 4 @ set fp to point to the base of the saved regs

___ ___, ___, ___ @ allocate space for the above local vars
```

Write the appropriate ARM instructions to perform the following. Treat every statement separately and independently. Do not assume any value in any register from a previous statement is available in a later stmt.

```c
*(a + 3) = a[1];

ptr = &a[0];

++ptr; // same as ptr = ptr + 1;

*ptr = a[2]; @ NOTE: Use r2 for ptr and r3 for a[2]
```

Use magic numbers and direct offsets from the frame pointer vs. assembler constants or other calculations.

Use register r3 as an intermediate / scratchpad register for all of these instructions. All local var accesses must be done via fp (not sp).

What is the expected return value in the following unit test cases from PA1 such that TEST will print PASSED?

```
TEST( checkRange( 1, 2, 1 ) == _____ );
TEST( isOdd( -11 ) != _____ );
```

Give the equivalent C pointer expression for the following array expression assuming bar is defined as an array. For this answer it does not matter what type each array element is or how many elements are defined.

```
bar[42] is equivalent to ____________________________ (equivalent pointer expression using no [ ]).
```