#1. a) Write the SPARC assembly instructions to define the following **global** variables in the **data** segment:

```assembly
char Yahoo[] = "Hack Me!";
short stuff = -37;
```

#2. What is the value (in hex) of %o1 after each set of instructions:

a) 
```assembly
set 0xFACEBEAD, %o1
sra %o1, 12, %o1
```
Value in %o1 at this point is **0x**

b) 
```assembly
set 0xFACEBEAD, %o1
set 0x3C3CACDC, %o2
xor %o1, %o2, %o1
```
Value in %o1 at this point is **0x**

c) 
```assembly
set 0xFACEBEAD, %o1
set 0x3C3CACDC, %o2
and %o1, %o2, %o1
```
Value in %o1 at this point is **0x**

#3. Assume you run gdb on pa1.
State how to set a breakpoint at the entry point in displaySquare():

Assume you correctly set this breakpoint and performed a run with correct command line arguments.
State how to print the value of the 2\textsuperscript{nd} argument passed to displaySquare() in gdb:
#4. Write the equivalent **unoptimized** SPARC assembly language instructions to perform the following C code fragment. **Use the loop construct specified in class/Notes.**

```c
a = 5678; /* x is mapped to %l2 */
/* a is mapped to %l4 */
while ( a > 181 )
{
    x = a - 37;
    a = a + 83;
}
```

#5a. Write the equivalent **unoptimized** SPARC assembly language instructions to perform the following C code fragment.

```c
x = x / 9876; /* x is mapped to %l0 */
```

5b. Now optimize your answer from #5a to eliminate any delay slots:

**Optimized version of above SPARC assembly**