Local variables and calling convention

Answer the following questions.

1.
```
#include <stdio.h>

int main()
{
    int a = 10;
    {
        int a = 20;
        {
            int a = 30;
            {
                int a = 40;
                printf("&a: %p a:%d\n", &a, a);
            }
            printf("&a: %p a:%d\n", &a, a);
        }
        printf("&a: %p a:%d\n", &a, a);
    }

    {
        int a = 20;
        {
            int a = 30;
            {
                int a = 40;
                printf("&a: %p a:%d\n", &a, a);
            }
            printf("&a: %p a:%d\n", &a, a);
        }
        printf("&a: %p a:%d\n", &a, a);

    }
    printf("&a: %p a:%d\n", &a, a);
    return 0;
}
```

Consider the above program. Let us consider that the address of the stack at the start of `main` is "X". What will be the addresses of all local variables in the `main` routine? You have to use the local variable allocator

discussed in the class (i.e., all local variables are allocated at function entry and deallocated just before the function return). [10]

2. Now compile and run the above routine using `gcc`. What do you think, how `gcc` is doing the local variable allocation and deallocation? [10]

3. Give an example in which moving a local variable allocation and deallocation to its outer scope (when the outer scope is not outside the function body) is much more efficient than allocating and deallocating within the scope. [10]

4. Compile "CallConv.c" using ``` ``gcc -fno-stack-protector -O1 CallConv.c'' ```. Dump the assembly into a file "filename.txt" using ``` ``objdump -dx a.out > filename.txt'' ```. Inspect the disassembly and answer the following questions. [32]

   - How arguments are passed to `f1` (i.e., which register/memory location is used to pass an argument).
   - How the return value of `f2` is returned to `main`.
   - How the return value of `f3` is returned to `main`.
   - How arguments are passed to `f4`.

How to submit

Submit your handwritten answers in the class after the lecture.