4. Consider the following two blocks of code, found in separate files:

```c
/* main.c */
int i = 0;
int main()
{
    foo();
    return 0;
}

/* foo.c */
int i = 1;
void foo()
{
    printf("%d", i);
}
```

What will happen when you attempt to compile, link, and run this code?

(a) It will fail to compile.
(b) It will fail to link.
(c) It will raise a segmentation fault.
(d) It will print “0”.
(e) It will print “1”.
(f) It will sometimes print “0” and sometimes print “1”.

5. The function `bitsy` is declared in C as

```c
int bitsy(int x);
```

and the (correctly) compiled IA32 code is:

```
bitsy: push %ebp
    mov %esp, %ebp
    sub $0x8, %esp
    mov 0x8(%ebp), %eax
    not %eax
    inc %eax
    leave
    ret
```

What is the result (denoted here by a C expression) returned by `bitsy`?

(a) !(x + 1)
(b) *(1 - x)
(c) -x
(d) (x > 0 ? -x: -x + 1)
Problem 3. (3 points):
Consider the executable object file a.out, which is compiled and linked using the command:

```
unix> gcc -o a.out main.c foo.c
```

and where the files main.c and foo.c consist of the following code:

```c
/* main.c */
#include <stdio.h>
static int a = 1;
int b = 2;
int c;
int main()
{
    int c = 3;
    foo();
    printf("a=%d, b=%d, c=%d", a, b, c);
    return 0;
}

/* foo.c */
int a, b, c;
void foo()
{
    a = 4;
    b = 5;
    c = 6;
}
```

What is the output of a.out?

a=_____, b=_____, c=____

Problem 4. (2 points):
The declaration of myGlobal in the following code is not executable, while the declaration of myLocal is executable, that is, the compiler does not emit any instructions to be executed at run time for the former, while it does for the latter. Briefly explain why (for both cases), and how myGlobal is initialized if there are no run time instructions for it.

```
int myGlobal = 10 + 7;

int mySub() {
    int myLocal = 0;
    ...
}
```

Answer:
Problem 4. (4 points):
The assembly code for a C function is given below.

```assembly
foo3:
    pushl %ebp
    movl %esp, %ebp
    movl 12(%ebp), %ecx
    movl 16(%ebp), %edx
    leal (%edx,%ecx,), %eax
    imull %ecx, %edx
    imull 8(%ebp), %eax
    popl %ebp
    addl %edx, %eax
    ret
```

Write the corresponding C function.

Problem 5. (4 points):
Consider the following program, which consists of two modules:

```c
/* Module foo6.c */
void p2(void);

int main() {
    p2();
    return 0;
}

/* Module bar6.c */
#include <stdio.h>
char main;
void p2()
{
    printf("0x%x\n", main);
}
```

When this program is compiled and executed, it prints the string "0x55\n" and terminates normally, even though `p2` never initializes variable `main`. Can you explain this?
Problem 6. (6 points):
Consider the following two .c files which both include the same .h file.

/* a.h */

int inc(int x) { return x + 1; } /* (1) */
int x; /* (2) */
int z; /* (3) */
void b(void); /* (4) */

/* a.c */
#include "a.h"
#include <stdio.h>
int w; /* (5) */
v = 5; /* (6) */

int main()
{
    x = inc(0);
z = 4;
b();
    printf("x = %d z = %d \\
"w = %d v = %d
", \\
x, z, w, v);
    return 0;
}

/* b.c */
#include "a.h"
int w = 4; /* (7) */
v = 5; /* (8) */

void b() {
    x = inc(x);
z--;
v++;
}

When compiled, linked, and executed, the following output results:

$ gcc a.c b.c -o a
$ ./a
x = 1 z = 3 w = 4 v = 5

Assuming that this program compiled and linked successfully, and based on the output show above, add static and/or extern modifiers to the blank lines (1) through (8)! If neither modifier would be appropriate, then write “no modifier” next to the comment. Any correct solution that results in successful compilation and the output shown will be accepted.