Important note: You must use the following retlib.c and exploit_1.c files to complete the assignment, and not the programs in the document above; otherwise, no credit will be given. You also need to read sections 3.1 and 3.2 to complete the assignment.

retlib.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int bof(FILE *badfile)
{
        char buffer[48];
        /* The following statement has a buffer overflow problem */
        fread(buffer, sizeof(char), 76,  badfile);
        return 1;
}

int main(int argc, char **argv)
{
        FILE *badfile;
        badfile = fopen("badfile", "r");
        bof(badfile);
        printf("Returned Properly\n");
        fclose(badfile);
        return 1;
}
```

exploit_1.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char ** argv)
{   char buf[76];

   FILE *badfile;
   badfile = fopen("badfile", "w");

   /* You need to decide the address and the values for X, Y, Z. The

        order of the three statements does not imply the order of X, Y, Z.

        Actually, we intentionally scrambled the order. */

   *(long *) &buf[X] = some address; //"/bin/sh"

   *(long *) &buf[Y] = some address; // system()

   *(long *) &buf[Z] = some address; // exit()


   fwrite(buf, sizeof(buf), 1, badfile);
   fclose(badfile);
}
```

Note: You can use the following syntax to specify addresses in exploit_1.c and exploit_2.c:

```
0xb3ea81f0
```

The 0x prefix implies that the address (b3dea81f0) is given in hexadecimal. (0xb3dea81f0 is a random address.)