```
//is_elf() determines if the file is ELF or not
check whether the first 4 bytes correspond to magic number(0x7f, “E”, “L”, “F”) //

bool is_elf(Elf64_Ehdr eh )
{
    return eh.e_ident[EI_MAG0] == ELFMAG0 && eh.e_ident[EI_MAG1] == ELFMAG1 &&
    eh.e_ident[EI_MAG2] == ELFMAG2 && eh.e_ident[ELFMAG2] &&
    eh.e_ident[EI_MAG3] == ELFMAG3;
}

void print_section_headers(int32_t fd, Elf64_Ehdr eh, Elf64_Shdr sh_table[])
{
    uint32_t i;
    char* sh_str;
    char* ro;
    char* modify;
    char* find = "rodata rodata rodata Can you modify this?";

    assert(lseek(fd, (off_t)eh.e_shoff, SEEK_SET) == (off_t)eh.e_shoff);

    for(i=0; i<eh.e_shnum; i++) {
        assert(read(fd, (void *)&sh_table[i], eh.e_shentsize) == eh.e_shentsize);
}

/* section-header string-table */
    sh_str = read_section(fd, sh_table[eh.e_shstrndx]);

    for(i=0; i<eh.e_shnum; i++) {
        if(!strncmp((sh_str + sh_table[i].sh_name), ".rodata", 7)) {
            ro = read_section(fd, sh_table[i]); //pointer to the buffer containing .rodata contents
            ro = realloc(ro, sh_table[i].sh_size +1); //realloc buffer to have size of .rodata + 1 (to make
space for "0" at the end)
            ro[sh_table[i].sh_size] = 0; //set last element of buffer 0
            char* p = ro;

            while (p < ro+sh_table[i].sh_size) { //iterate p as long as it does not go over .rodata size
                if ((modify = strstr(p,find)) != NULL) { //if we can locate the find string
                    memset(modify, 0, strlen(find)); //set the substring pointed by modify to be 0
                    memcpy(modify, "I modified it", strlen("I modified it")); //replace substring pointed
by modify to "I modified it"
                    assert(lseek(fd, (off_t)sh_table[i].sh_offset, SEEK_SET) == (off_t)
(sh_table[i].sh_offset)); //go to position of .rodata
                    assert(write(fd, ro, sh_table[i].sh_size) == sh_table[i].sh_size); //write contennt of
buffer ro to .rodata
}
                p += strlen(p) + 1; //if substring not found, increment p to the next string

}
            free(ro);
}
```

```
}
    free(sh_str);

}
```