Assignment #8a

Compiling and Debugging C Programs

Congratulations! You've inherited another program written by someone else. This one is buggy and broken.

1) Download the program `letters.c` into your workspace, and then unpack it as follows:

```
tar xvzf ex8a.tgz
cd ex8a
```

2) Create a "make" file to help compile your program, just as you did in Assignment #7.

3) Use "make" to compile your program. Wow, there are a lot of errors! The C compiler tells you the line number for each error. Open the program code in your favorite editor, and look for the first syntax error. Fix the error, save the code, and try compiling again. Continue this process until the file compiles cleanly.

4) Now, try to run the program. If it's working correctly, you should see something like this:

```
make
gcc -g letters.c -o letters –lm
./letters
Type in a sentence:
Hello, world!
Statistics: 2 words

Letter d: 1
Letter e: 1
Letter h: 1
Letter l: 3
Letter o: 2
Letter r: 1
Letter w: 1
```

But this program is buggy, so when you run it, it just seems to hang. What is it doing?

5) Use `gdb` to debug the program. Tell `gdb` to "run" the program, then press *Ctrl+C* to interrupt it and see where it's getting hung up:

```
gdb letters
(gdb) run
Starting program: letters
^C
Program received signal SIGINT, Interrupt.
main (argc=1, argv=0xbffff354) at letters.c:18
18        for (i=0; i < NLETTERS; i) {
(gdb)
```

6) Use `gdb` to step through the execution of the code and figure out all of the problems. Once you've fixed the bug that makes it hang up, give it a sentence and check the letter counts. Is it working correctly? If not, then step through your program and figure out why. Keep fixing the program until it works correctly.

Lessons Learned:

- A program must be compiled before running it, after each change to the source code.
- Even if a program compiles correctly, it doesn't necessarily run correctly.
- You can't tell what's happening inside a program unless you use `gdb` to step through it.