

Lotsa Letters.

This week we are going to count letters in a file. There are a lot of letters in this file, so we can't just have one input statement per letter. This would make our program too big, and would make our life very boring. So we need to read a file and test it to determine whether we have read everything. We need to execute the same code over and over to accomplish this.

We will be using two new things. The first is testing a file to see if you're at the end. This is done like this.

```
MyFile.eof()
```

This function returns TRUE if we are at end of file, and FALSE otherwise. There is a trick to this. If our file contains ten letters we need to read the file ELEVEN times to detect end of file. The ELEVENTH read will fail and cause MyFile.eof() to return true.

The next thing we need is a statement that allows us to execute the same code over and over again. Here it is.

```
while (<condition>
{
    <Lots of statements>
}
```

While the <condition> is true, the <Lots of statements> will be executed over and over. You read every character in a file by doing this.

DO IT THIS WAY!

```
MyFile>>Mychar; // read first character
```

```
while (!MyFile.eof())
{
    <Do Some Stuff>
    MyFile>>MyChar; // read the next character – must be at end of while.
}
```

Each time you read you will get one of the letters, a, b, c, d, e, f, or g. They will be lowercase. Count all of the a's all of the b's, ... etc. and compute a grand total.

See the other side for the required output format.

Print the following report:

```
57 characters total
12 A's
19 B's
5 C's
3 D's
2 F's
11 G's
5 other things
```

The grand total at the top should be the sum of the 8 other numbers. Just in case I accidentally slipped something other than a, b, c, d, e, f, or g into the input data, make sure to have an “other” category which is everything else, and print this at the end.

Make sure to read characters like this:

```
MyFile>>MyChar;
```

If you use other input functions you might read in some garbage characters. The file has a number of line-breaks which should be ignored and the >> operator will automatically ignore them for you.

There are lots of ways to read files. You may know of some other ways to do it. **HOWEVER**, if you want full credit for this program **YOU MUST DO IT THE WAY IT IS SHOWN IN THIS HANDOUT**.

Recommendation:

Use the “switch” statement to determine which character you’ve read.

```
switch (MyChar)
{
    case 'a':
        ...
        break;
    case 'b':
        ...
        break;
    ...
    default: ...
}
```