Some Common Mistakes

This list is based off examples I have seen from students' code as well as examples that the TAs have seen.

1) `if (ans == 'D' || 'd')`

Or connects two boolean expressions. For the example above, theses two expressions are

> **a)** `ans == 'D'`
> **b)** `'d'`

The first evaluates to either 0 or 1, based upon what is stored in ans. The second is a character literal. This particular character has an ascii value of 100, so the second expression ALWAYS evaluates to 100, which is ALWAYS TRUE.

What should have been typed is the following:

```
if (ans == 'D' || ans == 'd')
```

2) Missing brackets

This is self-explanatory. When you want more than one statement in a construct, you must place braces to indicate the beginning and end of the block within the construct.

One good technique to prevent this from being a problem is ALWAYS placing matching braces for ALL constructs before typing any code inside of the constructs, like so:

```
while (ans != 'Q') {

}
```

3) `value = value;`

While this statement will do NO harm, it won't do any good either. All this statement does, is evaluate whatever is stored in the variable value, and then stores that value back into value. It just stores the same thing into value, so no effect occurs from this statement.

4) `while (ans != 'Q' || ans != 'q')`

This statement ALWAYS evaluates to true. If ans stores 'Q', then the first part of clause is false, but the second part is TRUE! If ans stores 'q', then the first part of the clause is TRUE. Basically, this is always true because ans can't be equal to two different values at the same time!!!

Here's what should be done:

```
while (ans != 'Q' && ans != 'q')
```

5) `if (a > b > c)`

The way this is interpreted is as follows:

```
((a > b) > c)
```

The expression a > b is evaluated and will always return either a 0 or a 1, for true or false. THEN this value, 0 or 1, will be compared with c. Sure, it will evaluate to something, but it won't be what was logically intended. Here's how it should have been written:

```
if ((a > b) && (b > c))
```

6) `if (ans == D)`

This is an error if ans is a variable of type char and we want to see if it stores the character literal 'D'. Character literals are designated by single quotes. A literal is a single value where as a variable stores a value, and that value can change from time to time, but a literal does not change.

There's a key difference between doing something like:

```
if (value == 2)
```
and
```
if (value == number)
```

Namely, number could stand for any integer, whatever happens to be stored inside of it. But, 2 can only stand for 2. See, you would never think of declaring:

```
int 4;
```

so you shouldn't try to declare

```
char D;
```

If you just wanted to refer to the character 'D'.
Similarly, string literals are always between double quotes.