Goal
Understand if-statements and if-else statements.

if-statements

All programming languages provide constructs to execute a section of code conditionally. In the C programming language the simplest such statement has the following form:

```c
if (condition) {
    one-or-more statements
}
```

When program execution reaches the if-statement, the condition is evaluated. If the condition is true, the body of the if-statement is executed. If it false, the body of the if-statement is ignored. For example, the following code prints a message only if the value of n is less than two, and otherwise does nothing. The use of curly braces is not a requirement. However, if the braces are omitted, the compiler assumes that there is only one statement in the body of the if. For example,

```c
if (n < 2) {
    printf("Hello\n");
}
```

always executes the second printf statement, regardless of the value of n.

if-else statements

Another construct is called the if-else statement, and consists of two parts. If the condition is true, the first part is executed and the second part is ignored. If the condition is false, then the first part is ignored and the second part is executed. Note exactly one part is always executed.
if (condition) {
    one-or-more statements
}
else {
    one-or-more-statements
}

In the following example, for what values of \( n \) does the code produce the string “yes” and for what values of \( n \) does the code produce the string “no”?

```c
if ( (2 < n) && (n < 6) ) {
    printf("yes\n");
} else {
    printf("no\n");
}
```

Omitting braces can cause an unexpected error. For example, what does the following code fragment output?

```c
int n = 3;
if (n < 4)
    printf("hello\n");
else
    printf("good\n");
    printf("bye\n");
```

**cascading if-else**

Often an if-else is nested within the else part of an if-else statement, resulting in something which is logically like the following:

```c
if (condition) {
    statements
}
else {
    if (condition) {
        statements
    } else {
        statements
    }
}
```
For readability, some of the braces and indentation is suppressed resulting in the following form. Note the nesting can be done arbitrarily deep. Moreover, the final else is optional.

```c
if (condition) {
    statements
}
else if (condition) {
    statements
}
else if (condition) {
    statements
} else {
    statements
}
```

**dangling else problem**
The compiler associates an else-part with the closest if. The following code illustrates this point. What does it print? Remember that the compiler ignores formatting.

```c
int n = 5;
printf("hello\n");
if (n < 4)
    if (n > 0)
        printf("good\n");
    else
        printf("bye\n");
```

**Assignment:**

Write a complete C language program named lab4.c that will use fscanf to read one integer value named count. Don’t forget to prompt the user with instructions for input before using a scanf so that the user knows what the program is waiting for. The program then will read and count the number of integer scores.

We will assume scores at least 90 earn an “A”, scores less than 90 but at least 80 earn a “B”, and all other scores earn “C”. After reading all integer scores, your program should print how many were “A”, “B”, “C”.

**Turn In Work**

Show your TA that you completed the assignment. Then turn in your lab4.c program using the handin page: http://handin.cs.clemson.edu