Syntax and conditions

1)

```
    there are several ways to do this.  I will write all of the code
    in a single file.

#include <iostream>
using namespace std;
int main()
{
    int x, y;
    cin >> x >> y; //I didn't ask to prompt the user, but you might.

    //part a
    if(x > y)
    {
        cout << "x is larger!" << endl;
    }
    else if(y > x)
    {
        cout << "y is larger!" << endl;
    }
    else //they are equal
    {
        cout << "They are equal!" << endl;
        //note: I could have handled this case in many ways, but
        //this is what I chose.
    }
    //part b
    if(x < 0)
    {
        cout << "x is negative!" << endl;
    }
    if(y < 0) //use an if, not an else if, because I want two messages
    {
        cout << "y is negative!" << endl;
    }
    //part c
    if(y > x)
    {
        x = y;
    }
    //part d
    if(x > y)
    {
        y = 0;
    }
    //Note that for parts c and d, I didn't check for equality, since
    //the question is just looking for the case where one is LARGER
    //than the other

    return 0;
}
```

2)

First, we should be using a double or a float, since we are storing money, which can include decimals.

```
#include <iostream>
using namespace std;
int main()
{
    double preTaxAmount = 0.0;
    cout << "Enter the pre-tax amount";
    cin >> preTaxAmount;

    double finalAmount = preTaxAmount;

    if(preTaxAmount < 0)
    {
        cout << "Negative amounts not allowed!";
        return 0; //don't let anything else happen, just exit
    }
    else if(preTaxAmount > 10)
    {
        finalAmount = preTaxAmount * 1.13;
    }
    else if(preTaxAmount > 5)
    {
        //notice that I don't check for preTaxAmount < 10, since the
        //only way to get into this part of the code is if
        //preTaxAmount is less than 10.
        finalAmount = preTaxAmount * 1.07;
    }

    /*
        Did you notice that I didn't check for the case between 0 and
        5?  This is already handled when I declare finalAmount to be
        equal to preTaxAmount.  Since I don't need to change anything
        for this case, I didn't consider it.  If you did consider it,
        that's usually OK.  It just makes the code slightly longer.
    */

    //We didn't tell you to output, but we might as well
    cout << "The final amount is: " << finalAmount;
    return 0;
}
```

3)

```
#include <cmath>
#include <iostream>
using namespace std;
int main()
{
    double x = 0.0;
    cin >> x;

    /*
        I have two ways (at least...) to do this.  I will demonstrate
        both below.  In the first, I use only a single if statement,
        and I initialize the variable f_x to 1.  This avoids checking
        if x == 0, but also makes my code a bit less clear
    */
    double f_x = 1;
    if(x != 0)
    {
        f_x = sin(x)/x;
    }

    //here is the second way.  It's commented, so it won't run until
    //you uncomment.

    /*
    if(x == 0)
    {
        //it's a bit more clear that I have a special case for value
        //x = 0.
        f_x = 1;
    }
    else
    {
        f_x = sin(x)/x'
    }
    */
    cout << f_x;
    return 0;
}
```

4)

```
#include <cmath>
#include <iostream>
using namespace std;
int main()
{
    /*
        these variables should be doubles
        single letter names are appropriate here because of how the
        quadratic formula is normally taught.
    */
    double a, b, c;
    cin >> a >> b >> c;

    /*
        we'll compute the determinant.  It is more efficient,
        computationally, to store the value in a variable than to
        re-compute it.
    */
    double determ = pow(b,2.0) - 4*a*c;

    //now just check
    if(determ == 0)
    {
        cout << "One real root!";
    }
    else if(determ > 0)
    {
        cout << "Two real roots!";
    }
    else
    {
        cout << "No real roots";
    }
    return 0;
}
```

5)

Since the value 5 is greater than or equal to zero, we enter the second if statement. This decrements the variable to 4, which is the value we see on the screen.

Common Coding Errors

1)

For nearly all values of x, the left hand code and the right hand code are identical. The major issue is at the boundaries. There is one boundary case that should be considered: x = -1.

If you enter the value -1 to the left hand code:
The code executes the first if statement, since -1 is less than 0. At that point, the variable is incremented, and x now stores the value 0. Notice that the second if statement is just a pure if statement - it is not an else-if statement attached to that first if. As a result the *new* value of x is checked. This value is 0, so the second if statement is executed as well! In the second if statement, we decrement the value of x, so we will see -1 output to the screen.

If you enter the value -1 to the right hand code:
The code executes the first if statement, since -1 is less than 0. x is then incremented and now holds the value 0. Since the second if statement is actually an else if statement, it does NOT get executed, and the final output of the program is 0.

Since the two do not produce the same output for all inputs, the two code fragments are not identical.

2)

It is, in fact, possible. There are several ways to do it, but the method below does not introduce any new variables. Try to see if you can figure out how to do it by using additional variables instead of an additional if statement!

```
int x = 0;
cout << "Enter a value for x: ";
cin >> x;
if(x < 0 && x != -1)
{
    x = x + 1;
}
if(x >= 0)
{
    x = x - 1;
}
if(x == -1)
{
    x = x + 1;
}
cout << x;
```

Notice how the first if statement changed to handle the boundary case, but we weren't done. We had to account for that case. This had to be done *after* the second if statement (otherwise we'd be right back where we started). There is no clear reason why we would want to complicate the code in this manner when the else if statement was sufficient to capture the required functionality. As a result, it's important to write code that does not include redundant, complicated structure!

3)

For the first condition marked with FILL_IN_CONDITION_1, replace it with:

```
x < 10 && x > 5
```

Note that we have to check the condition x < 10 in this case, since this if statement will always check its conditions. Consider what happens if x = 20. The first if statement will execute, since 20>10. The second if statement *would* have executed if we were just checking for x > 5, since 20 > 5 is true as well.

For the second condition marked with FILL_IN_CONDITION_2, replace it with:

```
x < 5
```

We don't need to check for anything else, since anything that is less than 5 will also be less than numbers larger than 5.

4)

The main issue is that no matter which if statement executes, the line x = x + 1 is always executed, and it is always executed before any other code inside the if statements. We can remove it from all of the if statements and place it before the first one, and the code will produce the same output as before.

```
int x = 0;
cout << "Enter a value for x: ";
cin >> x;
x = x + 1;
if(x > 10)
{
    cout << “Big!”
}
else if(x > 5)
{
    cout << “Medium!”
}
else
{
    cout << “Small!”
}
cout << x;
```

5)

In this case, there is only one way for the output "REALLY big!" to be produced. x must be greater than 20. There is no reason to check if x > 10. The following code is identical in its functionality.

```
int x = 0;
cout << "Enter a value for x: ";
cin >> x;
if(x > 20)
{
    cout << “REALLY big!”
}
cout << x;
```