OUTPUT: (Assume b has address 66288)

\[ a = 2.0 \]
\[ b = 10 \]
\[ *p1 = 10 \]
\[ *p2 = 10 \]
\[ p1 = 66288 \]
\[ p2 = 66288 \]

REMARKS

- Observe that the * symbol has three distinct meanings
  - Value-at operator: \( *p1 \) above
  - Modification of data type into corresponding counter type: \( \text{int} * \)
  - Multiplication operator: \( 2 * 3 \)

- The address-of (\&) and value-at (\*) operators are in some sense inverses to one another, i.e. the expressions
  \[ \&a = a \quad \text{and} \quad a*pa = pa \]

Always evaluate to TRUE.
FUNCTIONS WITH OUTPUT PARAMETERS

There are two ways for a function to transfer data back to the calling function:

1. The function return value
2. Output parameters.

\[
\text{Return} \leftarrow \text{fcn}(\ldots, \ldots, \ldots, \ldots, \ldots, \ldots)
\]

\[
\begin{array}{c}
\text{VALUE} \\
\uparrow \quad \downarrow \\
\text{INPUT} \quad \text{OUTPUT} \quad \text{INPUT-OUTPUT} \\
\text{PARAM.} \quad \text{PARAM.} \quad \text{PARAM.}
\end{array}
\]

To create a function output parameter, we pass the address of a variable to the function. This gives the function the ability to alter the value of the variable, even though it is local to the calling function. This is done by assigning a value to the variable from within the function body, using the value-at (\&) operator.
Ex. (This one works!)

```c
void swap(int * pa, int * pb)
{
    int temp;
    temp = *pa;
    *pa = *pb;
    *pb = temp;
}

int main(void)
{
    int x = 2, y = 3;
    cout << "x = " << x << " y = " << y << endl;
    swap(&x, &y);
    cout << "x = " << x << " y = " << y << endl;
    return 0;
}
```

**OUTPUT**:

```
x = 2   y = 3
x = 3   y = 2
```

In this example, *pa and *pb are considered input-output parameters.
Ex. For use in circle2.cpp Example

```c
void find_area (double radius, double* parea)
{
    *parea = PI * radius * radius;
}
```

```c
int main (void)
{
    double a, r, c;

    find_area (r, &a);

    cout << "The area is " << a << endl;

    return (0);
}
```

In this example radius is an input parameter
while parea is an output parameter.

**Exercise:** Incorportate this function
into circle2.cpp.
```c
void order (double *pa, double *pb)
{
    double temp;
    if (*pa > *pb)
    {
        temp = *pa;
        *pa = *pb;
        *pb = temp;
    }
}

int main (void)
{
    double x = 20, y = 10, z = 30, w = 40;
    cout << "x = " << x << " y = " << y << " endl;"
    order (&x, &y);
    cout << "x = " << x << " y = " << y << " endl;"
    cout << "z = " << z << " w = " << w << " endl;"
    order (&z, &w);
    cout << "z = " << z << " w = " << w << " endl;"
    return 0;
}
```

Output:  
\[ x = 20 \quad y = 10 \]
\[ x = 10 \quad y = 20 \]
\[ z = 30 \quad w = 40 \]
\[ z = 30 \quad w = 40 \]
Review Fall Quiz 5

Trial Question #1:

```cpp
int x = 3, y = 2;

while (x != y)
{
    x++; 
    if (x > y)
        cout << "happy" << endl;
    else
        cout << "sad" << endl;
    x = 100; 
    y--; 
}
```

What is printed?
**Trick Question #1**

```cpp
int a = 5, b = 6, c = 7;

if (a == b)
    c = b + 2;
else
    c = b + 3;

cout << "happy" << endl;
if (b == c)
    cout << "sad" << endl;

a += b;

cout << "a = " << a << " b = " << b << " c = " << c << endl;
```

**What is printed?**