OUTPUT: (ASSUME b HAS ADDRESS 66288)

```
a = 20
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 POINTER TYPE: 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 AND &*Pa == Pa

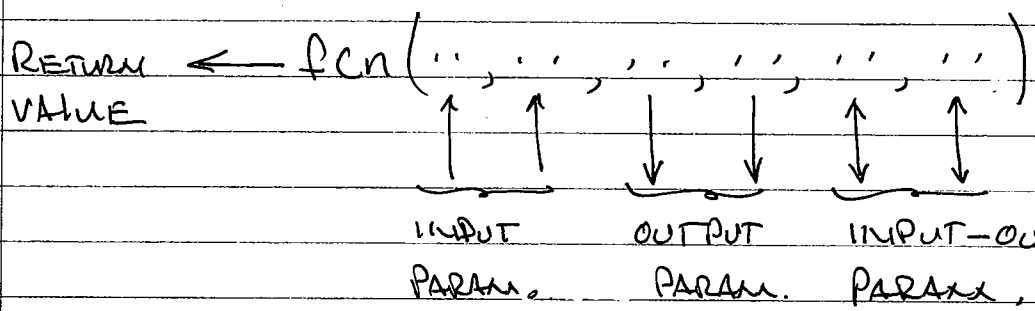
  ALWAYS EVALUATE TO TRUE.

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.



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!)

```
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

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

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: Incorporate this function into Circle2.cpp.

Ex.

```
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 y = 10
x = 10 y = 20
z = 30 w = 40
z = 30 w = 40

REVIEW FOR QUIZ 5

TRICK QUESTION #1:

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

while( x = y)
{
    x += 1;
    if ( x > y)
        cout << "happy" << endl;
    else
        cout << "sad" << endl;
        x = 100
    y--;
}
cout << "x=" << x << " y=" << y << endl;
```

WHAT is PRINTED?

TRICK QUESTION #2

```
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 ?