1) Give one example (JUST ONE) that will show that pass by value, pass by reference, pass by name, pass by value result are all different.

<table>
<thead>
<tr>
<th>a[0]</th>
<th>a[1]</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) Call-by-name  

b) Call-by-value  

c) Call-by-reference  

d) Call-by-value-result  

```cpp
int i, x;
int a[2];

void q(int b)
{
    a[0] = 4;
    i = 1;
    x = b;
    b = 0;
}

int main()
{
    i = 0; a[0] = 3; a[1] = 2;
    q(a[i]);
    cout << a[0] << " " << a[1] << " " << x << endl;
    return 0;
}
```

3) Here is a swap function in C++ that will swap two integers:

```cpp
void swap (int& a, int& b)
{
    int temp = a;
    a = b;
    b = temp
}
```

a) Given a language that only has pass by value and pass by name as parameter passing method. Write a swap function that will swap two integers. You cannot use pointers or object etc.
NOTE THAT WE CANNOT CHANGE THE PROTOCOL OF THE SWAP; IE
The header for swap must be: void swap(int x, int y);

int f ( int x, int y)
{
    int z = x;
    x = y;
    return z;
}
void swap (int x, int y)
{
    y = f(x,y);
}
void main()
{
    int a[]={1,2};
    i = 0
    swap(a[i], i);
    // or swap(i, a[i]);
}

b) Java only has pass by value, Write a swap function that will swap two integers.

    NOTE since java has only pass by value, you cannot swap two integers with the required swap protocol void swap(int x, int y);

    So we may try to wrap the integer using the Integer class:

    void swap (Integer a, Integer b)
    {
        Integer temp = a;
        a = b;
        b = temp;
    }

    int a = 1, b = 2;
    swap (a,b);

    THE ABOVE WILL NOT WORK ; draw some pictures and see.
Also, java Integers are immutable objects, hence you cannot change the Values of Integers.
So you have to write your own integer class.
class Int
{
    private int x;
    public Int(int i ) { x = i;}
    public void set(int i ) { x = i;}
    public int get() { return x;}
};

void swap(Int a, Int b)
{
    int temp = a.get();
    a.set(b.get());
    b.set(temp);
}

Hence there are many correct answers:
(1) No you cannot do it
(2) Use your own class.
(3) Use some other technique without following the protocol.

    static int swap(int a, int b) { return a;}

    public static void main(String args[])
    {
        int x = 10, y = 20;
        y = swap(x, x = y);
    }