Question: Consider the following class definition, and the sample output:

```cpp
#include <iostream>
using namespace std;
class Three_d
{
  int x, y, z;  // For holding 3-D coordinates
public:
  Three_d(){x=y=z=0;}
  Three_d(int i, int j, int k) {x=i; y=j; z=k;}

  // overloaded function declarations go here
  friend ostream& operator<<(ostream& o, const Three_d& a);
  const Three_d operator()(int a, int b, int c);
};

int main()
{
  Three_d obj1(1,2,3), obj2;
  obj2=obj1(10,11,12);  // overloading required here
  cout << "Obj1:"
  cout << obj1;        // overloading required here
  cout << endl;
  cout << "Obj2:"
  cout << obj2;        // overloading required here
  return 0;
}
```

The sample output is:

```
Obj1:1,2,3
Obj2:11,13,15
```

You have to overload two operators to allow the above code to execute without any errors and give the sample output. In the sample output obj2 is basically a sum of the original 3-D points of obj1 and the values provided within the ().

This means that original values of obj1 which are 1,2 and 3, are added to 10,11 and 12 respectively and assigned to obj2. The second operator to be overloaded is to enable `cout<<obj1;` to print the values of the object in this format: `1,2,3`

Provide implementations of the overloaded functions on the other side of this sheet.
```cpp
ostream& operator<<(ostream& o, const Three_d& a) {
    return (o << a.x << "," << a.y << "," << a.z);
}

const Three_d Three_d::operator()(int a, int b, int c) {
    Three_d temp;
    temp.x = x + a; // really means: temp.x = this->x + a
    temp.y = y + b;
    temp.z = z + c;
    return temp;
}
```