```
#include <iostream>

class Evaluable{
public:
 virtual double evaluate(double x)=0;
 virtual ~Evaluable()=default;
 virtual Evaluable* createCopy()=0;
};

class Sum : public Evaluable{
 Evaluable* e1;
 Evaluable* e2;

 void copy(const Sum& other) {
  e1 = other.createCopyE1();
  e2 = other.createCopyE2();
 }
 void move(Sum& other) {
  e1 = other.e1; e2 = other.e2;
  other.e1 = nullptr; other.e2 = nullptr;
 }

public:
 Sum(Evaluable* e1, Evaluable* e2): e1{e1}, e2{e2} {}
 Sum(const Sum& other) { this->copy(other); }
 Sum(Sum&& other) { this->move(other); }
 void operator=(const Sum& other) { delete e1; delete e2; this->copy(other); }
 void operator=(Sum&& other) { delete e1; delete e2; this->move(other); }
 virtual double evaluate(double x) { return e1->evaluate(x) + e2->evaluate(x); }
 virtual ~Sum() { delete e1; delete e2; }
 Sum* createCopy() { return new Sum(e1->createCopy(), e2->createCopy()); }
 Evaluable* createCopyE1() const { return e1->createCopy(); }
 Evaluable* createCopyE2() const { return e2->createCopy(); }
};


class Number : public Evaluable{
 double number;
public:
 Number(double number): number{number} {}
 Number(const Number& other) { number = other.number; }
 void operator=(const Number& other) { number = other.number; }
 virtual double evaluate(double x){ return number; }
 Number* createCopy() { return new Number(number); }
};

class X : public Evaluable{
public:
 virtual double evaluate(double x){ return x; }
 X* createCopy() { return new X(); }
};


int main(){
 auto E= new Sum(new Number(3), new Product(new Number(5), new Number(8)));
 for (int x=0; x<10; x++) std::cout << x << " " << E->evaluate(x) << std::endl;
 return 0;
}
```