```
/***********************************************************************
*****
* Author: Muhammad Rafi
*
* Purpose: Explaining Rule of Three (Example Class)
*
* Dated: August 18, 2007
*
* Version: 1.0
*                                                    )
* Last modified: August 18, 2007
*
***********************************************************************
****/
#include<iostream>
#include<cstdlib>
#include<new>

using namespace std;

class CAT{
 private:
     int *itsAge;
     int *itsWeight;
 public:
    CAT(); // Default constructor
    CAT(int a, int w); //Parameter Constructor
    CAT(const CAT & rhs); //Copy Constructor
    CAT & operator=(const CAT & rhs); //Assignment Operator
    char * Meow(); // Member function for supporting behaviour
    int getAge() const ; // Getter & Setter for Data Members
    int getWeight() const;
    void setAge(int a);
    void setWeight(int w);
    void * operator new (size_t size); // new for single object
    void operator delete (void *p); // delete for single object
    void * operator new[] (size_t size); // new for array of objects
    void operator delete[](void *p); // delete for array of objects
    ~CAT(); // Destructor
 };

CAT::CAT()
{
 itsAge= new int;
 itsWeight= new int;
 *itsAge=0;
 *itsWeight=0;

}

CAT::CAT(int a, int w)
{
      itsAge= new int;
 itsWeight= new int;
 *itsAge=a;
 *itsWeight=w;

}

CAT::~CAT()
```

```
{
    if (itsAge != 0)
    delete itsAge;
    itsAge=0;
     if (itsWeight != 0)
    delete itsWeight;
    itsWeight=0;
}

CAT::CAT(const CAT & rhs)
{
    itsAge= new int;
 itsWeight= new int;
 *itsAge=rhs.getAge();
 *itsWeight=rhs.getWeight();


}

CAT& CAT::operator=(const CAT & rhs)
{
     if (this==&rhs) return *this; // self assignment a=a
     else{
    itsAge= new int;
 itsWeight= new int;
 *itsAge=rhs.getAge();
 *itsWeight=rhs.getWeight();
      return *this;
               }
}


int CAT::getAge()const{ return *itsAge;}

int CAT::getWeight()const{ return *itsWeight;}

void * CAT::operator new (size_t size)
{     void *p;
      if( p= malloc(size)) return p; // memory for a single object
}// intentionally left for class users


void * CAT::operator new[] (size_t size)
{     void *p;
      if( p= malloc(size)) return p; // memory for array of objects
}// intentionally left for class users


void CAT::operator delete (void *p)
{
     if (p) free(p);
}

void CAT::operator delete[] (void *p)
{
     if (p) free(p);
}

int main()
{
    CAT myCat(2,3), yourCat(3,2);
    CAT tomCat(yourCat); // use of copy constructor
    CAT *CatCollection;
    CAT  *CatPtr;

    CatCollection = new CAT[10]; // use of new for array of object
    CatPtr = new CAT(); // use of new for a signle object

    CAT topCat= tomCat; // use of assignment operator


    cout<< myCat.getAge() <<endl;
    cout<< myCat.getWeight() <<endl;

    for (int i=0; i < 10 ; i++)
    {
        cout<<CatCollection[i].getAge() << endl;
        cout<<CatCollection[i].getWeight() << endl;
    }





    system("pause");



}
```