1]Plus Operator Overloading:

```
#include<iostream>
class Demo
{
    public:
        int a,b,c;

        Demo()
        {

        }

        Demo(int a,int b,int c)
        {
            this->a=a;
            this->b=b;
            this->c=c;
        }

        Demo* operator+(const Demo& ref)
        {
            Demo* temp=(Demo*)malloc(sizeof(temp));

            temp->a=this->a+ref.a;
            temp->b=this->b+ref.b;
            temp->c=this->c+ref.c;
            return temp;
        }

};

int main()
{
```

```
    Demo d1(3,5,6);

    Demo d2(5,6,7);

    Demo* d3=d1+d2;

    std::cout<<d3->a<<std::endl;
    std::cout<<d3->b<<std::endl;
    std::cout<<d3->c<<std::endl;
    return 0;
}
```

2]Minus Operator Overloading:

```
#include<iostream>
class Demo
{
    public:
        int a,b,c;

        Demo()
        {

        }

        Demo(int a,int b,int c)
        {
            this->a=a;
            this->b=b;
            this->c=c;
        }

        Demo* operator-(const Demo& ref)
        {
            Demo* temp=(Demo*)malloc(sizeof(temp));
```

```
            temp->a=this->a-ref.a;
            temp->b=this->b-ref.b;
            temp->c=this->c-ref.c;
            return temp;
        }

};

int main()
{
    Demo d1(3,5,6);

    Demo d2(5,6,7);

    Demo* d3=d1-d2;

    std::cout<<d3->a<<std::endl;
    std::cout<<d3->b<<std::endl;
    std::cout<<d3->c<<std::endl;
    return 0;
}
```

3]Multiplication Operator Overloading:

```
#include<iostream>
class Demo
{
    public:
        int a,b,c;

        Demo()
        {
```

```
        }

        Demo(int a,int b,int c)
        {
            this->a=a;
            this->b=b;
            this->c=c;
        }

        Demo* operator*(const Demo& ref)
        {
            Demo* temp=(Demo*)malloc(sizeof(temp));

            temp->a=this->a*ref.a;
            temp->b=this->b*ref.b;
            temp->c=this->c*ref.c;
            return temp;
        }

};

int main()
{
    Demo d1(2,6,4);

    Demo d2(5,6,7);

    Demo* d3=d1*d2;

    std::cout<<d3->a<<std::endl;
    std::cout<<d3->b<<std::endl;
    std::cout<<d3->c<<std::endl;
    return 0;
}
```

4]Division Operator Overloading:

```
#include<iostream>
class Demo
{
    public:
        int a,b,c;

        Demo()
        {

        }

        Demo(int a,int b,int c)
        {
            this->a=a;
            this->b=b;
            this->c=c;
        }

        Demo* operator/(const Demo& ref)
        {
            Demo* temp=(Demo*)malloc(sizeof(temp));

            temp->a=this->a/ref.a;
            temp->b=this->b/ref.b;
            temp->c=this->c/ref.c;
            return temp;
        }

};

int main()
{
    Demo d1(15,36,14);
```

```
    Demo d2(5,6,7);

    Demo* d3=d1/d2;

    std::cout<<d3->a<<std::endl;
    std::cout<<d3->b<<std::endl;
    std::cout<<d3->c<<std::endl;
    return 0;
}
```