```
#include<iostream>
using namespace std;
int Capital=50;  //global scope operator
class companyX{
        int Capital=50; //global variable
        int RAmount;
        public:
        void Require(int x){
                RAmount = x;
        }
        int Allocate(){
                Capital=Capital-RAmount;
                return Capital;
        }

//(1.1)Class variables are shared among instances
//(2.1)but can be overridden per instance,
//(1.2)while static variables belong strictly to the class and remain unchanged across instances.
//(2.2)Static variables do not allow per-instance modifications, whereas class variables can be shadowed by instance attributes.
};
int main(){
        companyX A,B,C;
        A.Require(15);
        int BalA=A.Allocate();
        B.Require(10);
        int BalB=B.Allocate();
        C.Require(9);
        int BalC=C.Allocate();
        cout<<BalA<<endl;
        cout<<BalB<<endl;
```

```
        cout<<BalC<<endl;
        return 0;
}
```

//gloabal variables equal to static variable

//difference between this is ()

//scope of a static variable is limited

//but the scope of a class is enitre the part

// static variable are accessble to all of the class (TRUE)

```
#include <iostudio>
using namespace std;

int name=79;//global scope operator

class variable{
        int name=79; //global variable
        int rlo;

        public:
        void require(int x){
                        rlo=x+x;
                }
        int allocate(){
                name=name/rlo;
                return name;
        }
};

int main(){
        variable x,y;
```

```
        a.require(20);
        int result1=a.allocate();
        cout<<"result2"<<result1<<endl;
  b.require(20);
        int result2=a.allocate();
        cout<<"result2="<<result2<<endl;
        return 0;
}
```