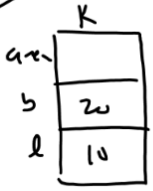
Solution 1

```
class Figure
{
   protected:
                 int l,b;
 public:
                 void get()
                 {
                   cout<<"Enter l,b:";
                   cin>>l>>b;
                 }
                void show()
                {
                   cout<<l<<","<<b<<endl;
                }          10        20
};
class Printdata
{
    public:
              void print(int a)
              {
                cout<<"Area is "<<a;
              }                   200
};
```



```
class Rectangle:public Figure,public Printdata
{
      int area;
public:
          void cal_area()
          {
             area=l*b;
             print(area);
          }                  20
};
int main()
{
✓  Rectangle R;
✓  R.get();
✓  R.show();
✓  R.cal_area();
   return 0;
}
```

Solution 2

```
class Figure
{
   protected:
                 int l,b;
 public:
                 void get()
                 {
                   cout<<"Enter l,b:";
                   cin>>l>>b;
                 }
                void show()
                {
                   cout<<l<<","<<b<<endl;
                }
};
class Printdata
{
    public:
              void print(int a)
              {
                cout<<"Area is "<<a;
              }
};
```

```
class Rectangle:public Figure,public Printdata
{
      int area;
public:
          int cal_area()
          {
             area=l*b;
             return area;
          }
};
int main()
{
   Rectangle R;
   R.get();
   R.show();
✓  int x=R.cal_area();
   R.print(x);
   return 0;
}
```

```
class Figure
{
   protected:
                  int l,b;
 public:
                  void get()
                  {
                    cout<<"Enter l,b:";
                    cin>>l>>b;
                  }
                 void show()
                  {
                    cout<<l<<","<<b<<endl;
                  }
};
class Printdata
{
    public:
              void show(int a)
              {
                cout<<"Area is "<<a;
              }
};
```

```
class Rectangle:public Figure,public Printdata
{
      int area;
public:
          int cal_area()
          {
              area=l*b;
              return area;

          }
};
int main()
{
   Rectangle R;
   R.get();
   R.show();
   int x=R.cal_area();
   R.show(x);
   return 0;
}
```

① Fig obj

② Fig *

③ Fig &

④ Drv obj

① Prnt ___

② P- - - -

③ R___

④ Drv obj

?

Error of Ambiguity

```
class Figure
{
   protected:
                  int l,b;
 public:
                  void get()
                  {
                    cout<<"Enter l,b:";
                    cin>>l>>b;
                  }
                 void show()
                  {
                    cout<<l<<","<<b<<endl;
                  }
};
class Printdata
{
    public:
              void show(int a)
              {
                cout<<"Area is "<<a;
              }
};
```

Solving Ambiguity problem in multiple inh

```
class Rectangle:public Figure,public Printdata
{
      int area;
public:
          int cal_area()
          {
              area=l*b;
              return area;

          }
};
int main()
{
   Rectangle R;
   R.get();
   R.Figure::show();
   int x=R.cal_area();
   R.Printdata::show(x);
   return 0;
}
```

Behaviour Of Constructor And Destructor In Inheritance

When we inherit a class and suppose the base and derived classes , both , have a constructor and destructor, then when we will create an object of derived class , the compiler will call BOTH THE CONSTRUCTORS but the order will be , from base to derived .

Similarly when the object of the derived class will be destroyed , then compiler will call the DESTRUCTOR of both the classes but the order will be from derived to base

```
#include <iostream>
#include <stdlib.h>
using namespace std;
class A
{
public:
   A()
   {
      cout<<"In constructor of base class A"<<endl;
   }
   ~A()
   {
      cout<<"In destructor of base class A"<<endl;
   }
};
class B:public A
{
public:
   B()
   {
      cout<<"In constructor of derived class B"<<endl;
   }
   ~B()
   {
      cout<<"In destructor of derived class B"<<endl;
   }
};
```

```
int main()
{
   B obj;
   return 0;
}
```

```
OUTPUT
========
In const of base class A
In const of derived class B
In dest of derived class B
In dest of base class A
```