1. Inheritence using access specifier

```
#include<bits/stdc++.h>
using namespace std;

class A{
   private:
      int x=10;
   protected:
      int y=20;
   public:
      int z=30;
};

class B : public A{
   public:
      B(){
         cout<<"From B x is private y = "<<y<<" z = "<<z<<endl;
      }
};

class C : protected A{
   public:
      C(){
         cout<<"From C x is private y = "<<y+10<<" z = "<<z<<endl;
      }
};

class D : private A{
   public:
      D(){
         cout<<"From D x is private y = "<<y<<" z = "<<z+10<<endl;
      }
};

int main(){
   B obj;
   C obj1;
   D obj2;
return 0;
}
```

2. **Diamond problem**

```
#include<bits/stdc++.h>
using namespace std;

class A{
   public:
      int i;
};

class B : virtual public A{
   public:
      int j;
};

class C : virtual public A{
   public:
      int k;
};

class D : public B, public C{
   public:
      int product(){
         return i*j*k;
      }
};

int main(){
   D obj;
   obj.i = 10;
   obj.j = 20;
   obj.k = 30;
   cout<<obj.product()<<endl;
return 0;
}
```

3. Parameterized constructor and multi-level inheritance

```
#include<bits/stdc++.h>
using namespace std;
```

```
class A{
   public:
      A(int a){
         cout<<a<<endl;
      }
};

class  B : public A{
   public:
      B(int a, int b):A(a){
         cout<<b<<endl;
      }
};

class C : public B{
   public:
      C(int a, int b, int c):B(a,b){
         cout<<c<<endl;
      }
};

int main(){
   C obj(1,2,3);
   return 0;
}
```

4. Speed compare using friend function

```
#include<bits/stdc++.h>
using namespace std;

class Truck;
class Car;
class Bike{
      int speed;
   public:
      Bike(int s){
         speed = s;
      }
      friend string sp_higher(Bike b, Car c, Truck t);
};
class Car{
      int speed;
```

```
    public:
        Car(int s){
            speed = s;
        }
        friend string sp_higher(Bike b, Car c, Truck t);
};

class Truck{
        int speed;
    public:
        Truck(int s){
            speed = s;
        }
        friend string sp_higher(Bike b, Car c, Truck t);
};

string sp_higher(Bike b, Car c, Truck t){
    if(b.speed > c.speed && b.speed > t.speed) return "Bike ";
    else if (b.speed < c.speed && c.speed > t.speed) return "Car ";
    else return "Truck ";
}

int main(){
    Bike b(100);
    Car c(120);
    Truck t(60);
    string h_speed = sp_higher(b,c,t);
    cout<<h_speed<<"is higher"<<endl;
return 0;
}
```

5. Default argument & function overloading

```
#include<bits/stdc++.h>
using namespace std;

void getResult(string id = "000000000"){
    cout<<id<<endl;
}

void getResult(int id, string dept=""){
    cout<<id<<" "<<dept<<endl;
```

```
}
void getResult(string id, string dept, string varsity=""){
   cout<<id<<" "<<dept<<" "<<varsity<<endl;
}
int main(){
   getResult();
   getResult("220201075");
   getResult("220201075", "CSE");
   getResult(220201075);
return 0;
}
```