Method Overriding in C++ - Problem Statements

Problem 1: Basic Method Overriding

Problem Statement: Create a base class `Animal` with a method `makeSound()`. Derive a class `Dog` that overrides the `makeSound()` method.

Sample Input:

```
Dog d;
d.makeSound();
```

Sample Output:

```
Woof Woof!
```

Problem 2: Overriding in a Hierarchical Inheritance

Problem Statement: Create a base class `Vehicle` with a method `fuelType()`. Derive `Car` and `Bike` classes that override `fuelType()`.

Sample Input:

```
Car c;
c.fuelType();
Bike b;
b.fuelType();
```

Sample Output:

```
Car uses petrol.
Bike uses diesel.
```

Problem 3: Virtual Function Demonstration

Problem Statement: Create a base class `Shape` with a virtual method `area()`. Derive `Rectangle` and `Circle` classes that override `area()`.

Sample Input:

```
Shape *s = new Rectangle(5, 10);
s->area();
```

Sample Output:

```
Area of rectangle: 50
```

Problem 4: Overriding with Complex Math Operations

Problem Statement: Create a base class `MathFunction` with a virtual method `compute()`. Derive `SquareRoot` and `Factorial` classes that override `compute()`.

Sample Input:

```
MathFunction *m = new SquareRoot(25);
m->compute();
```

Sample Output:

```
Square root: 5
```

Problem 5: Abstract Class with Pure Virtual Function

Problem Statement: Create an abstract class `Appliance` with a pure virtual method `powerConsumption()`. Derive `Fan` and `AC` classes that override `powerConsumption()`.

Sample Input:

```
Appliance *a = new Fan();
a->powerConsumption();
```

Sample Output:

```
Fan consumes 50W.
```

Problem 6: Overriding with Recursive Methods

Problem Statement: Create a base class `Sequence` with a virtual method `calculateNth()`. Derive `Fibonacci` and `Factorial` classes to compute the nth term recursively.

Sample Input:

```
Sequence *s = new Fibonacci(6);
s->calculateNth();
```

Sample Output:

```
Fibonacci(6): 8
```

Problem 7: Overriding with Constructors

Problem Statement: Create a base class `Instrument` with `playSound()`. Derive `Piano` and `Guitar` with appropriate overriding. Ensure base class constructor is called.

Sample Input:

```
Piano p;
p.playSound();
```

Sample Output:

```
Instrument created.
Playing piano.
```

Problem 8: Function Overriding in Multiple Inheritance

Problem Statement: Create two base classes `A` and `B` each having a method `show()`. Derive `C` and override `show()`.

Sample Input:

```
C c;
c.show();
```

Sample Output:

```
C's show() method.
```

Problem 9: Dynamic Method Dispatch with Pointers

Problem Statement: Create a base class `Game` with a virtual function `start()`. Derive `Chess` and `Football` that override `start()`. Use base class pointer to call overridden methods.

Sample Input:

```
Game *g = new Chess();
g->start();
```

Sample Output:

```
Starting Chess game.
```

Problem 10: Overriding with Cryptographic Algorithms

Problem Statement: Create a base class `Encryptor` with a virtual method `encryptData()`. Derive `AES` and `RSA` classes that override `encryptData()` to implement encryption algorithms.

Sample Input:

```
Encryptor *e = new AES("Hello");
e->encryptData();
```

Sample Output:

```
Encrypted data using AES.
---
```