SIMATS ENGINEERING

C++ Programming

P PRAVEEN
192110293

```cpp
#include <iostream>
#include <string>
class Animal {
public:
    Animal(const std::string& name) : name(name) {}
    virtual void speak() {
        std::cout << name << " makes a sound." << std::endl;
    }
    std::string getName() const {
        return name;
    }
private:
    std::string name;
};
class Cat : public Animal {
public:
    Cat(const std::string& name) : Animal(name) {}
    void speak() override {
        std::cout << getName() << " says 'Meow!'" << std::endl;
    }
};
class Dog : public Animal {
public:
    Dog(const std::string& name) : Animal(name) {}
    void speak() override {
        std::cout << getName() << " says 'Woof!'" << std::endl;
    }
};
```

Your input goes here. Give only values. Do not give like a=10

```
Fluffy says 'Meow!'
Rover says 'Woof!'
```

SIMATS ENGINEERING

C++ Programming

P PRAVEEN
192110293

```cpp
#include <iostream>
#include <cmath>
class Shape {
public:
    virtual double area() const = 0;
    virtual double volume() const = 0;
};
class Sphere : public Shape {
private:
    double radius;
public:
    Sphere(double r) : radius(r) {}
    double area() const override {
        return 4 * M_PI * radius * radius;
    }
    double volume() const override {
        return (4.0 / 3.0) * M_PI * std::pow(radius, 3);
    }
};
class Cylinder : public Shape {
private:
    double radius;
    double height;
public:
    Cylinder(double r, double h) : radius(r), height(h) {}
    double area() const override {
```

```
Your INPUT go's here! Give only values. do not give like a=10
```

```
Sphere Area: 113.097 sq units
Sphere Volume: 113.097 cubic units
Cylinder Area: 75.3982 sq units
Cylinder Volume: 50.2655 cubic units
```

SIMATS ENGINEERING

C++ Programming

P PRAVEEN
192110293

```cpp
#include <iostream>
class Shape {
public:
    virtual double area() const = 0;
    virtual double perimeter() const = 0;
};
class Rectangle : public Shape {
private:
    double length;
    double width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}
    double area() const override {
        return length * width;
    }
    double perimeter() const override {
        return 2 * (length + width);
    }
};
class Triangle : public Shape {
private:
    double base;
    double height;
    double side1;
    double side2;
public:
```

```
Your INPUT go's here! Give only values. do not give like a=10
```

```
Rectangle Area: 20 sq units
Rectangle Perimeter: 18 units
Triangle Area: 24 sq units
Triangle Perimeter: 20 units
```

```cpp
#include <string>
class Vehicle {
public:
    Vehicle(const std::string& name) : name(name) {}
    virtual void drive() {
        std::cout << name << " is being driven." << std::endl;
    }
    std::string getName() const {
        return name;
    }
private:
    std::string name;
};
class Car : public Vehicle {
public:
    Car(const std::string& name) : Vehicle(name) {}
    void drive() override {
        std::cout << getName() << " is being driven on the road." << std::endl;
    }
};
class Truck : public Vehicle {
public:
    Truck(const std::string& name) : Vehicle(name) {}
    void drive() override {
        std::cout << getName() << " is being driven on the highway." << std::endl;
    }
};
```

give like a=10

```
Sedan is being driven on the road.
Pickup Truck is being driven on the highway.
```

```
    virtual double calculatePay() {
        return baseSalary;
    }
    std::string getName() const {
        return name;
    }
protected:
    double baseSalary;

private:
    std::string name;
};
class Manager : public Employee {
public:
    Manager(const std::string& name, double baseSalary, double bonus) : Employee(na
    double calculatePay() override {
        return baseSalary + bonus;
    }
private:
    double bonus;
};
class Engineer : public Employee {
public:
    Engineer(const std::string& name, double baseSalary, double overtimePay) : Empl
    double calculatePay() override {
        return baseSalary + overtimePay;
```

```
Manager John earns: 60000
Engineer Alice earns: 65000
```

SIMATS ENGINEERING

C++ Programming

P PRAVEEN
192110293

```cpp
#include <iostream>
class Animal {
public:
    virtual void eat() {
        std::cout << "The animal is eating something." << std::endl;
    }
};
class Herbivore : public Animal {
public:
    void eat() override {
        std::cout << "The herbivore is eating plants." << std::endl;
    }
};
class Carnivore : public Animal {
public:
    void eat() override {
        std::cout << "The carnivore is eating other animals." << std::endl;
    }
};
int main() {
    Animal genericAnimal;
    Herbivore deer;
    Carnivore lion;
    genericAnimal.eat();
    deer.eat();
    lion.eat();
```

Your INPUT go's here! Give only values. do not give like a=10

```
The animal is eating something.
The herbivore is eating plants.
The carnivore is eating other animals.
```

SIMATS ENGINEERING

C++ Programming

P PRAVEEN
192110293

```cpp
#include<iostream>
class Person
{
  public:
  virtual  void work()
  {
    std::cout<<"person is workning"<<std::endl;
  }

};
class Employee:public Person
{
  public:

void work()
{
  std::cout<<"employee is working"<<std::endl;
}
};
class Manager:public Person
{
public:

  void work()
  {
    std::cout<<"manager is managing the team"<<std::endl;
  }
```

```
Your INPUT go's here! Give only values. do no
give like a=10
```

```
person is workning
employee is working
manager is managing the team
```

```
53 class Engineer : public Employee {
54
55 public:
56
57     Engineer(const std::string& name, double baseSalary, double overtimePay) : Empl
58
59     double calculatePay() override {
60
61         return baseSalary + overtimePay;
62
63     }
64
65 private:
66
67     double overtimePay;
68
69 };
70
71 int main() {
72
73     Manager manager("John", 50000.0, 10000.0);
74
75     Engineer engineer("Alice", 60000.0, 5000.0);
76
77     std::cout << "Manager " << manager.getName() << " earns: " << manager.calculate
78
```

```
Manager John earns: 60000
Engineer Alice earns: 65000
```