1. Person, Employee, and Manager Classes

```
#include <iostream>
using namespace std;
class Person {
public:
    virtual void work() {
        cout << "Person is working." << endl;
    }
};
class Employee : public Person {
public:
    void work() override {
        cout << "Employee is working on tasks." << endl;
    }
};
class Manager : public Person {
public:
    void work() override {
        cout << "Manager is managing the team." << endl;
    }
};
int main() {
    Person* p1 = new Employee();
    Person* p2 = new Manager();
    p1->work();
    p2->work();
    delete p1;
    delete p2;
    return 0;
}
```

Output

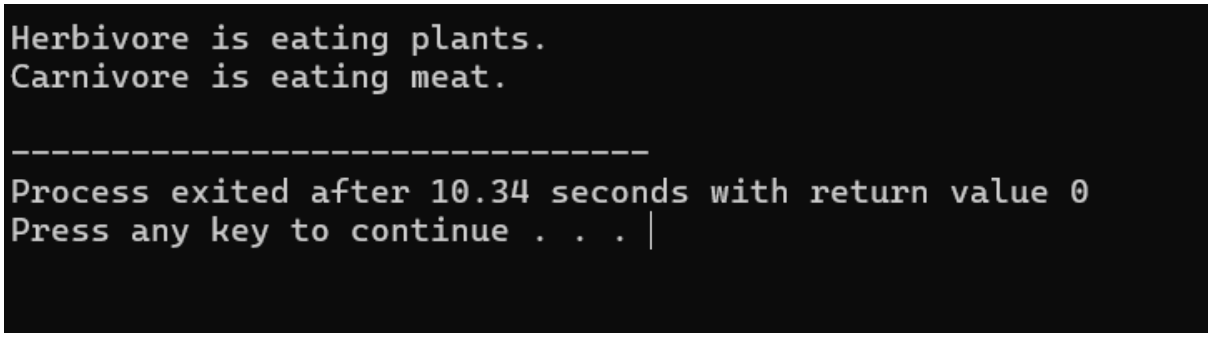
```
Employee is working on tasks.
Manager is managing the team.

--------------------------------
Process exited after 1.229 seconds with return value 0
Press any key to continue . . .
```

2. Animal, Herbivore, and Carnivore Classes

```
#include <iostream>
using namespace std;
class Animal {
public:
    virtual void eat() {
        cout << "Animal is eating." << endl;
    }
};
class Herbivore : public Animal {
public:
    void eat() override {
        cout << "Herbivore is eating plants." << endl;
    }
};
class Carnivore : public Animal {
public:
    void eat() override {
        cout << "Carnivore is eating meat." << endl;
    }
};
int main() {
    Animal* a1 = new Herbivore();
    Animal* a2 = new Carnivore();
```

```
    a1->eat();
    a2->eat();
    delete a1;
    delete a2;
    return 0;
}
```

Output

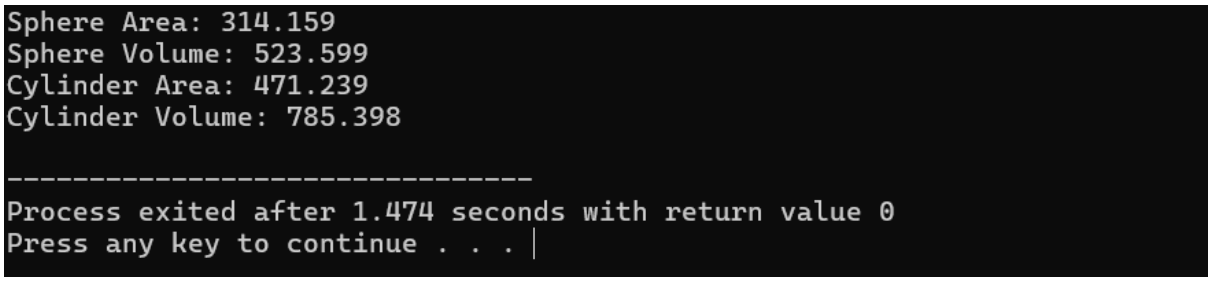


3. Shape, Sphere, and Cylinder Classes

```
#include <iostream>
#include <cmath>
using namespace std;
class Shape {
public:
    virtual double area() = 0;
    virtual double volume() = 0;
};
class Sphere : public Shape {
private:
    double radius;
public:
    Sphere(double r) : radius(r) {}
```

```
    double area() override {
        return 4 * M_PI * radius * radius;
    }
    double volume() override {
        return (4.0/3.0) * M_PI * radius * radius * radius;
    }
};
class Cylinder : public Shape {
private:
    double radius, height;
public:
    Cylinder(double r, double h) : radius(r), height(h) {}
    double area() override {
        return 2 * M_PI * radius * (radius + height);
    }
    double volume() override {
        return M_PI * radius * radius * height;
    }
};
int main() {
    Shape* s1 = new Sphere(5);
    Shape* s2 = new Cylinder(5, 10);
    cout << "Sphere Area: " << s1->area() << endl;
    cout << "Sphere Volume: " << s1->volume() << endl;
    cout << "Cylinder Area: " << s2->area() << endl;
    cout << "Cylinder Volume: " << s2->volume() << endl;
    delete s1;
    delete s2;
    return 0;
}
```

Output



4. Person, Student, and Teacher Classes (Greet Function)

```
#include <iostream>
using namespace std;
class Person {
public:
   virtual void greet() {
      cout << "Hello, I am a person." << endl;
   }
};
class Student : public Person {
public:
   void greet() override {
      cout << "Hello, I am a student." << endl;
   }
};
class Teacher : public Person {
public:
   void greet() override {
      cout << "Hello, I am a teacher." << endl;
   }
};
int main() {
   Person* p1 = new Student();
```

```
    Person* p2 = new Teacher();
    p1->greet();
    p2->greet();
    delete p1;
    delete p2;
    return 0;
}
```

Output

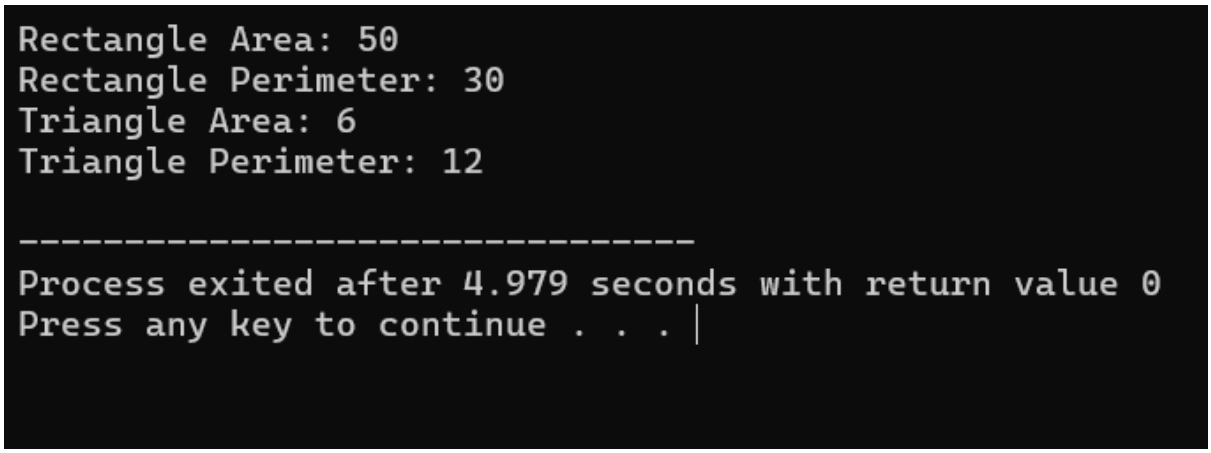
```
Hello, I am a student.
Hello, I am a teacher.

--------------------------------
Process exited after 3.553 seconds with return value 0
Press any key to continue . . .
```

5. Shape, Rectangle, and Triangle Classes (Area and Perimeter)

```
#include <iostream>
#include <cmath>
using namespace std;
class Shape {
public:
    virtual double area() = 0;
    virtual double perimeter() = 0;
};
class Rectangle : public Shape {
private:
    double width, height;
public:
    Rectangle(double w, double h) : width(w), height(h) {}
    double area() override {
        return width * height;
```

```
    }
    double perimeter() override {
        return 2 * (width + height);
    }
};
class Triangle : public Shape {
private:
    double a, b, c;
public:
    Triangle(double side1, double side2, double side3) : a(side1), b(side2), c(side3) {}
    double area() override {
        double s = (a + b + c) / 2;
        return sqrt(s * (s - a) * (s - b) * (s - c));
    }
    double perimeter() override {
        return a + b + c;
    }
};
int main() {
    Shape* s1 = new Rectangle(5, 10);
    Shape* s2 = new Triangle(3, 4, 5);
    cout << "Rectangle Area: " << s1->area() << endl;
    cout << "Rectangle Perimeter: " << s1->perimeter() << endl;
    cout << "Triangle Area: " << s2->area() << endl;
    cout << "Triangle Perimeter: " << s2->perimeter() << endl;
    delete s1;
    delete s2;
    return 0;
}
```

Output



6. Vehicle, Car, and Truck Classes (Drive Function)

```
#include <iostream>
using namespace std;
class Vehicle {
public:
   virtual void drive() {
      cout << "Vehicle is driving." << endl;
   }
};
class Car : public Vehicle {
public:
   void drive() override {
      cout << "Car is driving on the road." << endl;
   }
};
class Truck : public Vehicle {
public:
   void drive() override {
      cout << "Truck is driving on the highway." << endl;
   }
```

```
};
int main() {
    Vehicle* v1 = new Car();
    Vehicle* v2 = new Truck();
    v1->drive();
    v2->drive();
    delete v1;
    delete v2;
    return 0;
}
```

Output

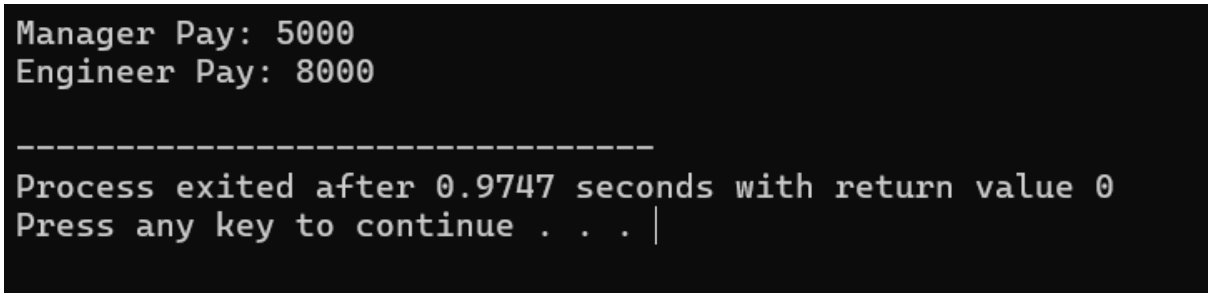
```
Car is driving on the road.
Truck is driving on the highway.

--------------------------------
Process exited after 12.3 seconds with return value 0
Press any key to continue . . .
```

7. Employee, Manager, and Engineer Classes (Calculate Pay Function)

```
#include <iostream>
using namespace std;
class Employee {
public:
    virtual double calculatePay() = 0;
};
class Manager : public Employee {
private:
    double salary;
public:
    Manager(double sal) : salary(sal) {}
```

```
    double calculatePay() override {
        return salary;
    }
};
class Engineer : public Employee {
private:
    double hourlyRate;
    int hoursWorked;
public:
    Engineer(double rate, int hours) : hourlyRate(rate), hoursWorked(hours) {}
    double calculatePay() override {
        return hourlyRate * hoursWorked;
    }
};
int main() {
    Employee* e1 = new Manager(5000);
    Employee* e2 = new Engineer(50, 160);
    cout << "Manager Pay: " << e1->calculatePay() << endl;
    cout << "Engineer Pay: " << e2->calculatePay() << endl;
    delete e1;
    delete e2;
    return 0;
}
```

Output



8. Animal, Cat, and Dog Classes (Speak Function)

```
#include <iostream>
using namespace std;
class Animal {
public:
    virtual void speak() {
        cout << "Animal is making a sound." << endl;
    }
};
class Cat : public Animal {
public:
    void speak() override {
        cout << "Cat says meow." << endl;
    }
};
class Dog : public Animal {
public:
    void speak() override {
        cout << "Dog says woof." << endl;
    }
};
int main() {
    Animal* a1 = new Cat();
    Animal* a2 = new Dog();
    a1->speak();
    a2->speak();
    delete a1;
    delete a2;
    return 0;
}
```

Output

```
Cat says meow.
Dog says woof.

--------------------------------
Process exited after 1.156 seconds with return value 0
Press any key to continue . . .
```

9. Shape, Rectangle, and Circle Classes (Area Function)

```
#include <iostream>
#include <cmath>
using namespace std;
class Shape {
public:
   virtual double area() = 0;
};
class Rectangle : public Shape {
private:
   double width, height;
public:
   Rectangle(double w, double h) : width(w), height(h) {}
   double area() override {
      return width * height;
   }
};
class Circle : public Shape {
private:
   double radius;
public:
   Circle(double r) : radius(r) {}
```

```
    double area() override {
        return M_PI * radius * radius;
    }
};
int main() {
    Shape* s1 = new Rectangle(5, 10);
    Shape* s2 = new Circle(7);
    cout << "Rectangle Area: " << s1->area() << endl;
    cout << "Circle Area: " << s2->area() << endl;
    delete s1;
    delete s2;
    return 0;
}
```

Output