Run Save

```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 {
        double lateralSurfaceArea= 2 * M_PI * radius * height;
        double baseArea = M_PI * radius * radius;
        return lateralSurfaceArea + 2 * baseArea;
```

```
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

K.HEMANTH
192110453

Run Save

```
#include<iostream>
using namespace std;
class Person
  {
     public:
           virtual void work()
             {
               cout<<"person is working"<<endl;
             }
  };
class Employee: public Person
{
     public:
           void work()
           {
              cout<<"employee is working"<<endl;
           }
   };
class Manager: public Person
{
     public:
           void work()
           {
              cout<<"manager is working"<<endl;
           }
  };
int main()
  {
    Person person1,*p1,*p2,*p3;
```

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

```
person is working
employee is working
manager is working
```

C++ Programming - IDE

Not secure | 172.18.34.31/cpp_program/onlineide.php

SIMATS ENGINEERING

C++ Programming

K.HEMANTH
192110453

Run Save

```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:
    Triangle(double b, double h, double s1, double s2) : base(b), height(h), side1(s1), side2(s2) {}
    double area() const override {
        return 0.5 * base * height;
```

```
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
```

Type here to search

Google Chrome

15:06
13-10-2023

SIMATS ENGINEERING

C++ Programming

K.HEMANTH
192110453

Run | Save | Changes Updated, Saved 53

```cpp
#include <iostream>
#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;
    }
};
int main() {
```

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

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

C++ Programming - IDE

Not secure | 172.18.34.31/cpp_program/onlineide.php

SIMATS ENGINEERING

C++ Programming

K.HEMANTH
192110453

Run | Save | Changes Updated, Saved 51

```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;
    }
};
int main() {
```

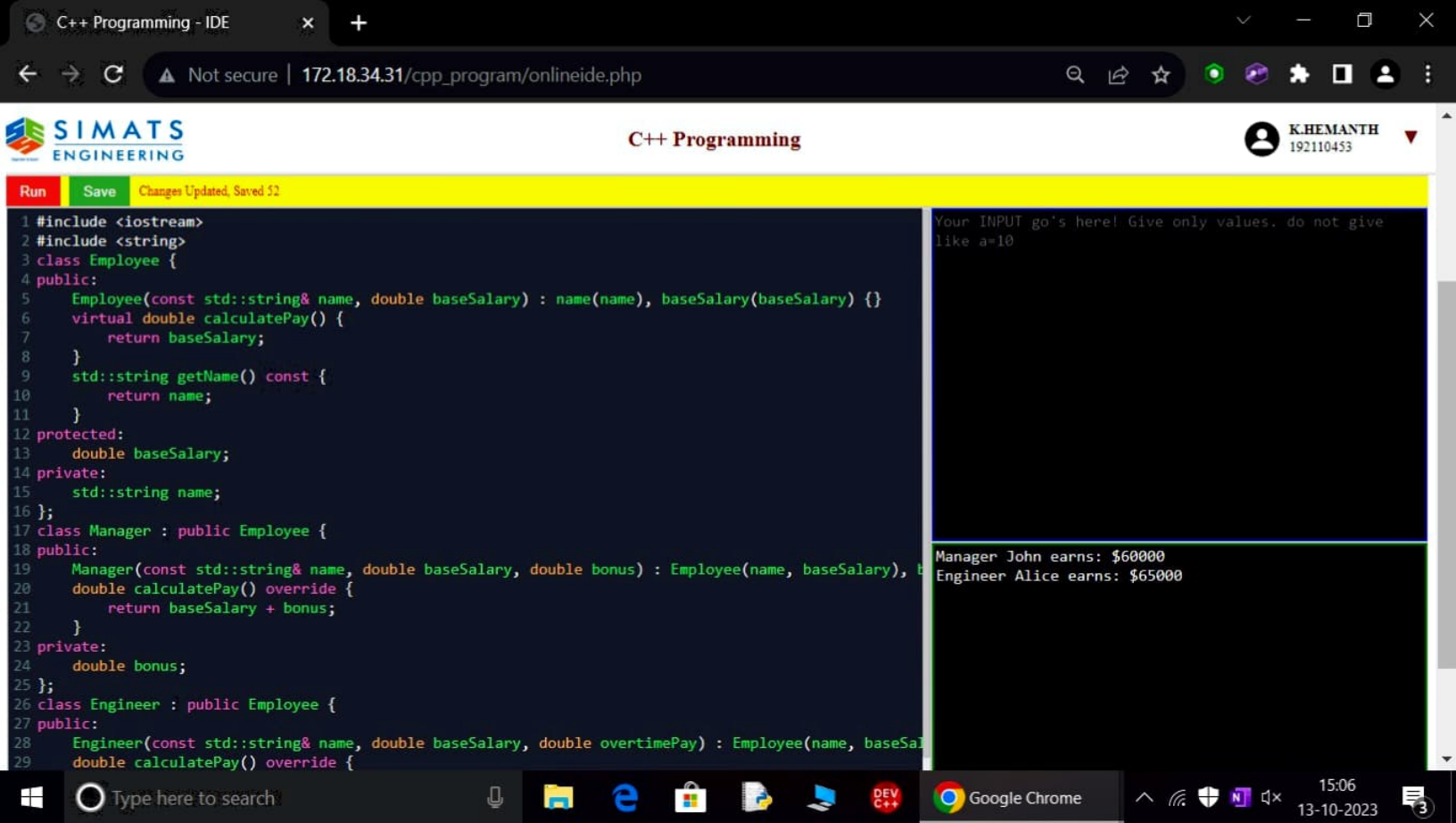
Your INPUT go's here! Give only values. do not give like a=10

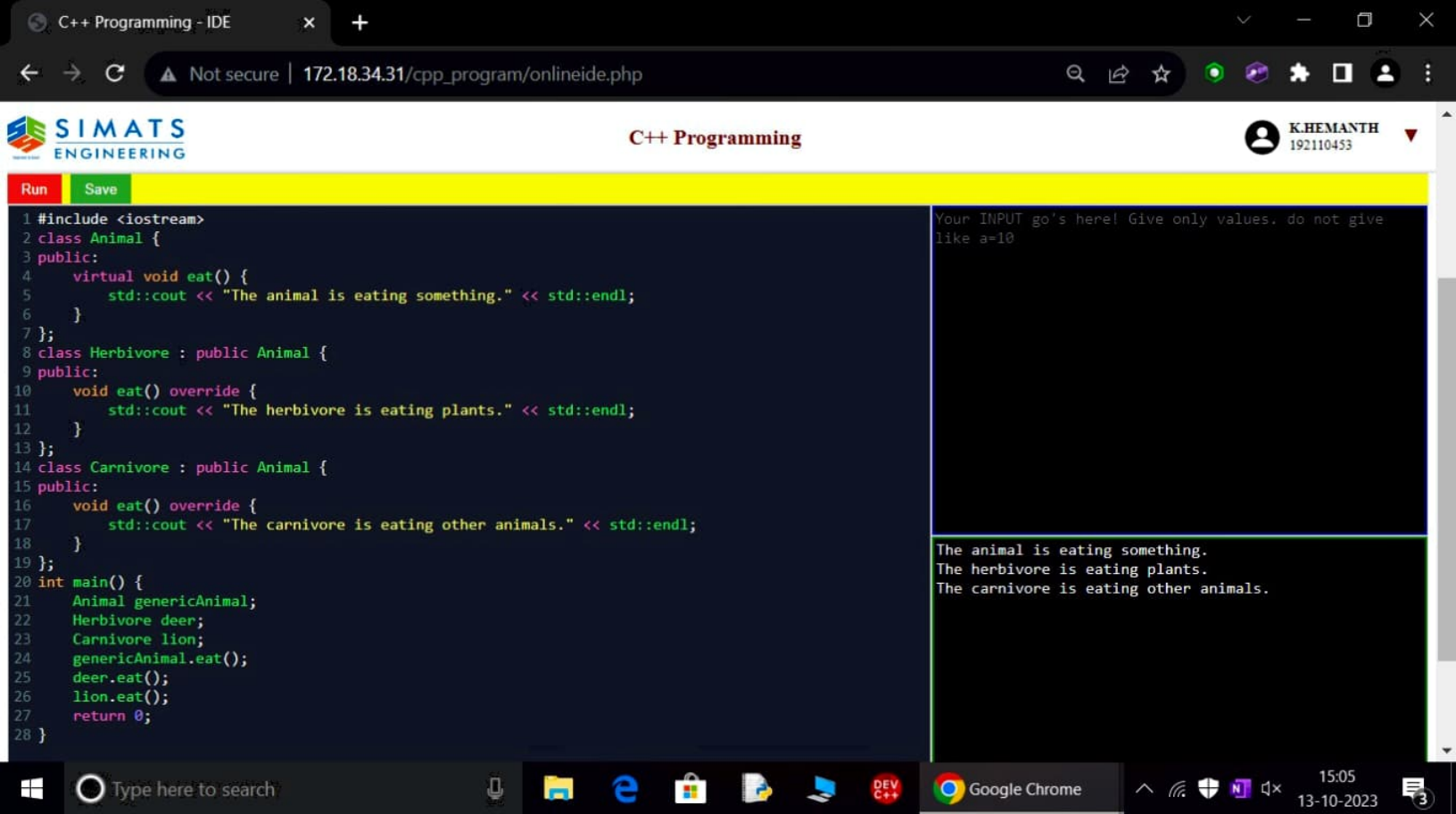
```
Fluffy says 'Meow!'
Rover says 'Woof!'
```

Type here to search

Google Chrome

15:06
13-10-2023


C++ Programming - IDE
Not secure | 172.18.34.31/cpp_program/onlineide.php
SIMATS ENGINEERING
C++ Programming
K.HEMANTH
192110453
Run
Save
Changes Updated, Saved 52
#include <iostream>
#include <string>
class Employee {
public:
Employee(const std::string& name, double baseSalary) : name(name), baseSalary(baseSalary) {}
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(name, baseSalary), b
double calculatePay() override {
return baseSalary + bonus;
}
private:
double bonus;
};
class Engineer : public Employee {
public:
Engineer(const std::string& name, double baseSalary, double overtimePay) : Employee(name, baseSal
double calculatePay() override {
Your INPUT go's here! Give only values. do not give like a=10
Manager John earns: $60000
Engineer Alice earns: $65000
Type here to search
Google Chrome
15:06
13-10-2023


C++ Programming - IDE
Not secure | 172.18.34.31/cpp_program/onlineide.php
SIMATS ENGINEERING
C++ Programming
K.HEMANTH
192110453
Run
Save
#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();
return 0;
}
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.
Type here to search
Google Chrome
15:05
13-10-2023