/\* Write a C++ program by creating an 'Employee' class having the following functions and print the final salary.

1 - 'getInfo()' which takes the salary, number of hours of work per day of employee as parameters

2 - 'AddSal()' which adds $10 to the salary of the employee if it is less than $500.

3 - 'AddWork()' which adds $5 to the salary of the employee if the number of hours of work per day is more than 6 hours.

Input constraints:

- The initial salary of the employee is an integer between 0 and 1000 (inclusive).

- The number of work hours per day is an integer between 1 and 10 (inclusive).

Output constraints:

- The final salary of the employee is an integer between 0 and 1000 (inclusive). \*/

#include <iostream>  
using namespace std;  
class employee  
{  
    double salary;  
    int no\_of\_hours;  
public:  
    employee() {}  
    void getinfo()  
    {  
        cout<<"Enter the salary of employee:"<<endl;  
        cin>>salary;  
        cout<<"Enter the number of hours:"<<endl;  
        cin>>no\_of\_hours;  
    }  
    void AddSal()  
    {  
        if (salary<500){  
            salary=salary+10;  
        }  
    }  
    void AddWork()  
    {  
        if (no\_of\_hours>6){  
            salary=salary+5;;  
        }  
    }  
    void DisplaySal()  
    {  
        cout<<salary;  
    }  
};  
int main()  
{  
    int n;  
    cout<<"Enter the number of employees:"<<endl;  
    cin>>n;  
    employee\*emp=new employee[n];  
    for (int i=0;i<n;i++){  
        emp[i].getinfo();  
        emp[i].AddSal();  
        emp[i].AddWork();  
    }  
    for (int i=0;i<n;i++){  
        cout<<"\nThe final salary of employee "<<i<<" is:"<<endl;  
        emp[i].DisplaySal();  
    }  
}

/ WAP to calculate volume of a cube using constructor  
#include <iostream>  
class cube  
{  
 private:  
 double side\_length;  
 public:  
 //Constructor  
 cube(double length){  
 side\_length=length;  
 }  
 //To calculate volume  
 double calculate\_volume(){  
 return side\_length\*side\_length\*side\_length;  
 }  
};  
int main()  
{  
 double length;  
 std::cout<<"Enter length of side of cube:";  
 std::cin>>length;  
 //Create a cube object  
 cube cube(length);  
 //Calculate volume  
 double volume=cube.calculate\_volume();  
 std::cout<<"Volume of cube is:"<<volume<<std::endl;  
 return 0;  
}

// WAP to calculate volume of a cube using constructor  
#include <iostream>  
class rectangle  
{  
 private:  
 double length;  
 double width;  
 public:  
 //Constructor  
 rectangle(double len,double wid){  
 length=len;  
 width=wid;  
 }  
 //To calculate area  
 double calculate\_area(){  
 return length\*width;  
 }  
};  
int main()  
{  
 double length,width;  
 std::cout<<"Enter length of rectangle:";  
 std::cin>>length;  
 std::cout<<"Enter width of rectangle:";  
 std::cin>>width;  
 //Create a rectangle object  
 rectangle rectangle(length,width);  
 //Calculate volume  
 double area=rectangle.calculate\_area();  
 std::cout<<"Area of rectangle is:"<<area<<std::endl;  
 return 0;  
}

/\* WAP to create a BankAccount class in C++ that stimulates a bank account. It should have methods to withdraw,deposit and check account balance. Use necessary constructors and accessors. \*/

#include <iostream>

using namespace std;

class BankAccount

{

private:

double balance;

public:

BankAccount (){

balance=0;

}

void deposit(double amount)

{

balance=balance+amount;

}

void withdraw(double amount)

{

if (amount>balance)

{

cout<<"Insufficient Balance";

}

else

{

balance=balance-amount;

}

}

double getBalance()

{

return balance;

}

};

int main() {

BankAccount acc;

int n;

double amt;

while (true)

{

amt=0;

cout<<"\nTo Deposit money press 1.\nTo Withdraw money press 2.\nTo Display balance press 3.\nTo Quit press 4."<<endl;

cin>>n;

if (n==1)

{

cout<<"Enter the Amount to deposit"<<endl;

cin>>amt;

acc.deposit(amt);

}

else if(n==2)

{

cout<<"Enter the Amount to Withdraw"<<endl;

cin>>amt;

acc.withdraw(amt);

}

else if (n==3)

{

cout<<"Balance is:"<<acc.getBalance()<<endl;

}

else if(n==4)

{

break;

}

}

return 0;

}

/\* Create a class called Stack that represents a stack data structure in C++ using array. It should have methods push(), pop() and isEmpty(). WAP to demonstrate functionality of Stack class by performing push and pop operations. \*/

#include <iostream>

using namespace std;

class Stack {

private:

static const int MAX\_SIZE = 100;

int data[MAX\_SIZE];

int top;

public:

Stack() {

top = -1;

}

bool isEmpty() const {

return top == -1;

}

bool isFull() const {

return top == MAX\_SIZE - 1;

}

void push(int element) {

if (isFull()) {

cout << "Stack is full. Cannot push element." << endl;

} else {

top++;

data[top] = element;

cout << "Pushed element: " << element << endl;

}

}

void pop() {

if (isEmpty()) {

cout << "Stack is empty. Cannot pop element." << endl;

} else {

int poppedElement = data[top];

top--;

cout << "Popped element: " << poppedElement << endl;

}

}

};

int main() {

Stack stack;

int choice;

int element;

do {

cout << "Stack Operations:" << endl;

cout << "1. Push" << endl;

cout << "2. Pop" << endl;

cout << "3. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter element to push: ";

cin >> element;

stack.push(element);

break;

case 2:

stack.pop();

break;

case 3:

cout << "Exiting program." << endl;

break;

default:

cout << "Invalid choice. Please try again." << endl;

break;

}

cout << endl;

} while (choice != 3);

return 0;

}

// There is a base class named shape. This base class has features such as length, breadth and height. There are sub classes namely square, rectangle, triangle, circle and parallelogram. Inherit the base class in the required places and implement a function to find the area of the shapes. Use constructors. 1 sq m = Rs 50.

#include <iostream>  
using namespace std;  
class Shape  
{  
 protected:  
 double length;  
 double breadth;  
 double height;  
 public:  
 Shape(double l=0,double b=0,double h=0): length(l),breadth(b),height(h){}  
 virtual double calculateArea()=0;  
};  
class Square:public Shape  
{  
 public:  
 Square(double side):Shape(side,side,0){}  
 double calculateArea() override{  
 return length\*length;  
 }  
};  
class Rectangle:public Shape  
{  
 public:  
 Rectangle(double l,double b):Shape(1,b,0){}  
 double calculateArea() override{  
 return length\*breadth;  
 }  
};  
class Triangle:public Shape  
{  
 public:  
 Triangle(double b,double h):Shape(0,b,h){}  
 double calculateArea() override{  
 return 0.5\*breadth\*height;  
 }  
};  
class Circle:public Shape  
{  
 public:  
 Circle(double radius):Shape(radius,0,0){}  
 double calculateArea() override{  
 return 3.14159\*length\*length;  
 }  
};  
class Parallelogram:public Shape  
{  
 public:  
 Parallelogram(double b,double h):Shape(0,b,h){}  
 double calculateArea() override{  
 return breadth\*height;  
 }  
};  
int main()  
{  
 Shape\*shapes[]={  
 new Square(5),  
 new Rectangle(4,6),  
 new Triangle(3,8),  
 new Circle(2),  
 new Parallelogram(4,5),  
 };  
 double totalArea=0;  
 for(Shape\*shape:shapes){  
 double area=shape->calculateArea();  
 totalArea=totalArea+area;  
 cout<<"Area:"<<area<<"sq.m"<<endl;  
 }  
 double costPerSqM=50;  
 double totalCost=totalArea\*costPerSqM;  
 cout<<"Total cost: Rs"<<totalCost<<endl;  
 for(Shape\*shape:shapes){  
 delete shape;  
 }  
 return 0;  
}