Shipping Calculator

Create a program that calculates the total cost of an order including the cost of items ordered and the shipping cost.

Note: **Bold** words are output while non-bold words are input in the following console samples.

Console Sample 1 (valid data)

```
Shipping Calculator

Cost of items ordered: 49.99

Shipping cost: 7.95
Total cost: 57.94
```

Console Sample 2 (invalid data 1)

```
Shipping Calculator

Cost of items ordered: -65.50

You must enter a positive number!
```

Console Sample 3 (invalid data 2)

```
Shipping Calculator

Cost of items ordered: 0

You must enter a positive number!
```

Specifications

- Use the following table to calculate shipping cost:

Range of Cost of Items Ordered	SHIPPING COST
0 < cost of items ordered < 30	5.95
30 <= cost of items ordered < 50	7.95
50 <= cost of items ordered < 75	9.95
cost of items ordered >= 75	0

- You have to define a class *Shipping* with one private data member corresponding to the cost of items ordered.
- Declare and define a non-default constructor which requires a parameter corresponding to the private data member for the class *Shipping*.
- Declare and define both the getter and setter for the private member in the class *Shipping*.
- Declare and define a public member function *get_shipping_cost()* in the class *Shipping* to calculate the shipping cost according to the table shown above by using if statements.
- Declare and define a public member function *get_total_cost()* in the class *Shipping* to calculate the total cost by using the formula:

```
total cost = cost of items ordered + shipping cost
```

- Declare and define a public member function *display_results()* in the class *Shipping* to display all the calculation results in the console as shown in the Console Sample 1 above by calling the other member functions *get_shipping_cost()* and *get_total_cost()*.
- In the *main()*,you must first determine the validity of the input of the cost of items ordered. If the cost of items ordered is positive, create a *Shipping* object based on it and then call the member function *display_results()* on the object; if the user enters a number that is less than or equal to zero, no object will be created and only display an error message as shown in the Console Sample 2 or 3 above: “You must enter a positive number!”
- The program should accept decimal entries like 35.5 and 14.25.
- There is no requirement of precision for the output.