

1. What does the following code do? Can it be used as a skeleton code which can then be modified and reused for solving different questions? Justify.

```
#include <iostream>

using namespace std;

class Sample {

public:

    void printText1();

    void printText2();

    void printValue(int value);

};

void Sample::printText1() {

    cout << "IncludeHelp.com\n"; }

void Sample::printText2() {

    cout << "Let's learn together\n"; }

void Sample::printValue(int value) {

    cout << "value is: " << value << "\n"; }

int main()

{

    Sample obj;

    obj.printText1();

    obj.printText2();

    obj.printValue(101);
```

```
return 0;  
}
```

Solution :

This code defines a class with methods to print text values.

Yes this can be used as a skeleton code because we can modify methods to fit different tasks .

2. Write a CPP program to design a calculator do the following:

(i) Add two natural numbers.

(ii) Add two complex numbers.

(iii) Add two matrices.

Your code must showcase the use of operator and/or function overloading.

```
#include <iostream>  
  
using namespace std;
```

```
int add(int a, int b) {  
    return a + b;  
}
```

```
class Complex {  
    private:  
        float real;
```

```
float imag;

public:

    Complex(float r = 0, float i = 0) : real(r), imag(i) {}


    Complex operator + (const Complex &c) {

        return Complex(real + c.real, imag + c.imag);

    }


    void display() const {

        cout << real << " + " << imag << "i" << endl;

    }

};
```

```
class Matrix {

private:

    int mat[2][2];

public:

    Matrix(int m[2][2]) {

        for (int i = 0; i < 2; ++i)

            for (int j = 0; j < 2; ++j)

                mat[i][j] = m[i][j];

    }

};
```

```

Matrix operator + (const Matrix &m) {
    int result[2][2];
    for (int i = 0; i < 2; ++i)
        for (int j = 0; j < 2; ++j)
            result[i][j] = mat[i][j] + m.mat[i][j];
    return Matrix(result);
}

```

```

void display() const {
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j)
            cout << mat[i][j] << " ";
        cout << endl;
    }
}

```

```
};
```

```
int main() {
```

```
    int num1 = 5, num2 = 10;
```

```
    cout << "Sum of natural numbers: " << add(num1, num2) << endl;
```

```
    Complex c1(3.4, 5.6), c2(1.2, 3.4);
```

```

Complex c3 = c1 + c2;

cout << "Sum of complex numbers: ";

c3.display();


int m1[2][2] = {{1, 2}, {3, 4}};

int m2[2][2] = {{5, 6}, {7, 8}};

Matrix mat1(m1), mat2(m2);

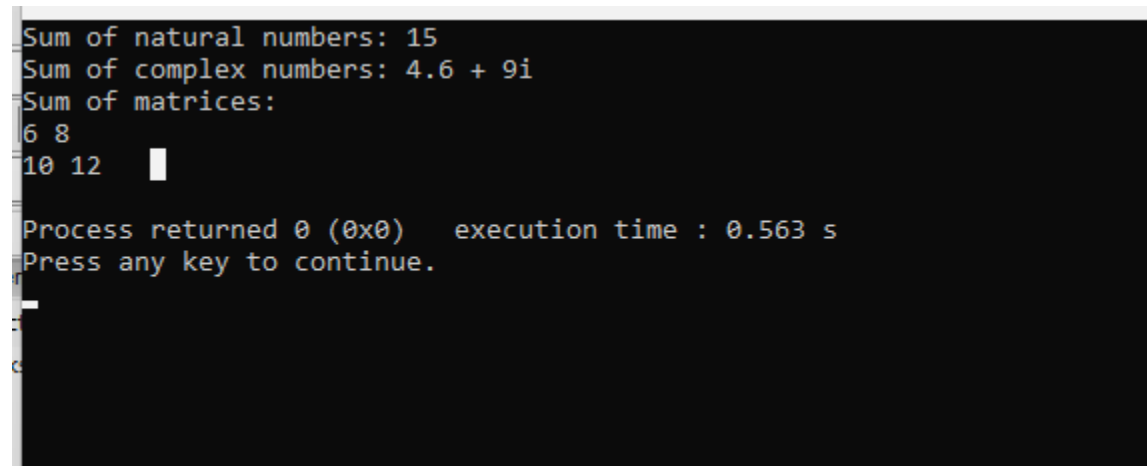
Matrix mat3 = mat1 + mat2;

cout << "Sum of matrices: " << endl;

mat3.display();


return 0;
}

```



```

Sum of natural numbers: 15
Sum of complex numbers: 4.6 + 9i
Sum of matrices:
6 8
10 12
Process returned 0 (0x0)   execution time : 0.563 s
Press any key to continue.

```

3. Write a CPP program to take input from user for 10 vendors who supply computer accessories. Design attributes and functions to satisfy the below mentioned

requirements. Write function to

(i) input accessory details of individual vendors.

(ii) print the above details.

(iii) Compare between the prices of the same component/accessory of different vendors.

(iv) Find the vendor who has maximum quantity of “LAN Cable” currently available.

(v) Find the vendor who has the lowest selling price of “Keyboard”.

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
const int n_vendors = 2;
```

```
class vendor {
```

```
public:
```

```
    string n;
```

```
    string acc;
```

```
    double p;
```

```
    int q;
```

```
    void in() {
```

```
    cout << "Enter vendor name: ";  
  
    getline(cin, n);  
  
    cout << "Enter accessory name: ";  
  
    getline(cin, acc);  
  
    cout << "Enter price of accessory: ";  
  
    cin >> p;  
  
    cout << "Enter quantity available: ";  
  
    cin >> q;  
  
    cin.ignore();  
  
}
```

```
void out() const {  
  
    cout << "Vendor: " << n << endl;  
  
    cout << "Accessory: " << acc << endl;  
  
    cout << "Price: $" << p << endl;  
  
    cout << "Quantity: " << q << endl;  
  
}  
};
```

```
void cmp_price(vendor v[], int n, const string& acc) {  
  
    double min_p = 1e10;  
  
    double max_p = -1;  
  
    string min_v, max_v;  
  
  
    for (int i = 0; i < n; ++i) {
```

```

if (v[i].acc == acc) {
    if (v[i].p < min_p) {
        min_p = v[i].p;
        min_v = v[i].n;
    }
    if (v[i].p > max_p) {
        max_p = v[i].p;
        max_v = v[i].n;
    }
}
}

```

```

if (min_p == 1e10) {
    cout << "Accessory not found." << endl;
} else {
    cout << "Lowest price for " << acc << " is $" << min_p << " by " << min_v << endl;
    cout << "Highest price for " << acc << " is $" << max_p << " by " << max_v << endl;
}
}

```

```

void max_lan(vendor v[], int n) {
    int max_q = 0;
    string v_name;

```



```

for (int i = 0; i < n; ++i) {

    if (v[i].acc == "LAN Cable" && v[i].q > max_q) {

        max_q = v[i].q;

        v_name = v[i].n;

    }

}

if (max_q == 0) {

    cout << "LAN Cable not found." << endl;

} else {

    cout << "Vendor with maximum quantity of LAN Cable is " << v_name << " with " << max_q << "
units." << endl;

}

}

```

```

void min_keyboard(vendor v[], int n) {

    double min_p = 1e10;

    string v_name;

    for (int i = 0; i < n; ++i) {

        if (v[i].acc == "Keyboard" && v[i].p < min_p) {

            min_p = v[i].p;

            v_name = v[i].n;

        }

    }

}

```

```

if (min_p == 1e10) {

    cout << "Keyboard not found." << endl;

} else {

    cout << "Vendor with the lowest price of Keyboard is " << v_name << " with price $" << min_p <<
endl;

}

}

```

```

int main() {

    vendor v[n_vendors];


    for (int i = 0; i < n_vendors; ++i) {

        cout << "Enter details for vendor " << (i + 1) << endl;

        v[i].in();

        cout << endl;

    }

```

```

    cout << "Vendor Details:" << endl;

    for (int i = 0; i < n_vendors; ++i) {

        v[i].out();

        cout << "-----" << endl;

    }

```

```

string acc;

cout << "Enter accessory name to compare prices: ";

getline(cin, acc);

cmp_price(v, n_vendors, acc);


max_lan(v, n_vendors);


min_keyboard(v, n_vendors);


return 0;
}

```

```

Enter details for vendor 1
Enter vendor name: abc
Enter accessory name: LABN Cable
Enter price of accessory: 300
Enter quantity available: 1

Enter details for vendor 2
Enter vendor name: xyz
Enter accessory name: LAN Cable
Enter price of accessory: 400
Enter quantity available: 1

Vendor Details:
Vendor: abc
Accessory: LABN Cable
Price: $300
Quantity: 1
-----
Vendor: xyz
Accessory: LAN Cable
Price: $400
Quantity: 1
-----
Enter accessory name to compare prices: LAN Cable
Lowest price for LAN Cable is $400 by xyz
Highest price for LAN Cable is $400 by xyz
Vendor with maximum quantity of LAN Cable is xyz with 1 units.

```

4. What will be the output of following C++ programs? Justify.

(a)

```
#include<iostream>

using namespace std;

class Test {

    int x;

};

int main()

{

    Test t;

    cout << t.x;

    return 0;

}
```

Error because "x" is private member of class and cannot be called outside class .

(b)

```
#include<iostream>

using namespace std;

class Empty {};

int main()

{

    cout << sizeof(Empty);

    return 0;

}
```

```
}
```

the answer will be 1 because the the empty class will have some addresss stored somewhere.

(c)

```
#include<iostream>

using namespace std;

class Test
{
    static int x;
    int *ptr;
    int y;
};

int main()
{
    Test t;
    cout << sizeof(t) << "\n ";
    cout << sizeof(Test *);
}
```

The output will be " 12 8 " because the size of pointer is 8 and int is 4 so the class will be of 12 and that of pointer to class will be 8 .

(e)

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
class Student {
```

```
private:
```

```
    int rollNo;
```

```
    string stdName;
```

```
    float perc;
```

```
public:
```

```
    void setValue()
```

```
{
```

```
    rollNo = 0;
```

```
    stdName = "None";
```

```
    perc = 0.0f;
```

```
}
```

```
    void printValue()
```

```
{
```

```
    cout << "Student's Roll No.: " << rollNo <<
```

```
    "\n";
```

```
    cout << "Student's Name: " << stdName <<
```

```
    "\n";
```

```
    cout << "Student's Percentage: " << perc <<
```

```
    "\n";
```

```

    }
};

int main()
{
    Student std;

    std.setValue();

    std.printValue();

    return 0;
}

```

Output will be :

0

None

0

Because the set functions sets values and then the same is printed using print function .

```

f)#include <iostream>

using namespace std;

class Person {};

int main() {

    Person per;

    cout <<< "size of per: " <<< sizeof(per) <<< endl;

    return 0;

}

```

Output will be size of per :1 because the class is empty .