Ganpat University

Faculty of Engineering & Technology

Computer Science & Engineering

Name:- Dwij Vatsal Desai

Sem:- 2

Sub: - ESFP-II

Enrollment No.:- 23162121027

Prac:- 12

Practical 12

Objective:

To learn about object-oriented programming, string, friend class and friend function, polymorphism (Function & Operator overloading) and inheritance.

Problem Definition: What will be the output of the following code? Give an explanation in brief.

Q1:

```
#include <iostream>
#include <string.h>
using namespace std;
class school
{
private:
    int roll_no;
string name;
 public:     void
display()
    {
        cout << "Enter roll__no and name=";
cin >> roll_no >> name;         cout <<
"\n"
            << roll_no << " " << name;
```

```
    }     friend class
student;
}; class
student
{     school
obj;
 public:
void show()
    {         obj.roll_no
= 1001;          obj.name =
"Rahul";          cout <<
"\n"
             << obj.roll_no << " " << obj.name;
    } };
int main()
{     school obj;
student obj2;
obj.display();
obj2.show();
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'q1.exe'
Enter roll__no and name=12 jaimin

12 jaimin
1001 Rahul
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - The student class has a friend relationship with the school class, but this is not used in the student class. Also, the show() function in the student class directly accesses the roll_no and name members of the obj object of the school class, which violates the encapsulation principle.

Q2:

```
#include <iostream>
using namespace std;
class invent1
{
    int code;
int items;
float price;
```

```
public:
    invent1() {}
invent1(int a, int b, int c)
    {
code = a;
items = b;
price = c;
    }     void
display()
    {         cout << "\nCode : " <<
code;         cout << "\nItems : " <<
items;         cout << "\nPrice : "
<< price;
    }     int
getcode()
    {
return code;
    }     int
getitem()
    {         return
items;
    }     int
getprice()
    {         return
price;
    }
};  class
invent2
{     int code;
float value;
// invent1 ob1;
public:
    invent1 ob1;
// ob1.display();
invent2()
    {
ob1.display();
code = 0;
value = 0;
    }     invent2(int x,
float y)
     {
code = x;
value = y;
    }
```

```
    void display()
    {         cout << "Code : " << code <<
endl;         cout << "Value : " << value
<< endl;
    }
invent2(invent1 p)
    {         code = p.getcode();
value = p.getitem() * p.getprice();
    } };
int main()
{     invent1 s1(100, 5, 140);     invent2 d1;     d1 = s1;
// Invoke Constructor in Invent2 for conversion     cout <<
"\n Product details - Invent1 type";     s1.display();
cout << "\n\n\n Product details - Invent2 type\n";
d1.display();     return 0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'

  Code : 0
  Items : 0
  Price : -2.21384e-10
   Product details - Invent1 type
  Code : 100
  Items : 5
  Price : 140


   Product details - Invent2 type
  Code : 100
  Value : 700
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - The program defines two classes: invent1 and invent2. invent1 represents an inventory item with code, items, and price. invent2 also represents an inventory item but calculates its value based on the items and price of invent1. In the main() function, an object of invent1 is created with specific values. Then, an object of invent2 is created and assigned the value of the invent1 object. When displaying the details of both objects, invent2 displays a value calculated from the items and price of invent1

Q3:

```
#include <iostream>
#include <string>
using namespace
std; class B {
    int b;
 public:
B() {}
    B(int i)
{
        b = i;
    }
    int show()
    {
return b;
    }
};
class C
{
B     b;

public:
    C(int i)
{         b =
B(i);
    }     friend void
show()
    {
C     c(10);         cout << "value of b is: " << c.b.show() <<
endl;
    } };
int main(int argc, char const *argv[])
{
    C c(1);
    c.show();
return 0;
}
```

Error - The error in the program is that the friend function show() in class C tries to access the private member b of class C, which is an object of class B, but show() is not a member function of class B. To fix this, either show() should be declared as a member function of class C, or it should be modified to take an object of class C as an argument to access its private members.

Q4:

```cpp
#include <iostream>
using namespace std;
class BaseA
{ public: BaseA() { cout << "BaseA
constructor called" << endl;
} };
class BaseB
{ public: BaseB() { cout << "BaseB
constructor called" << endl;
} }; class Derived: public BaseA, public BaseB
{ public: Derived() { cout << "Derived's
constructor called" << endl;
} }; int
main() {
Derived d;
return 0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
BaseA constructor called
BaseB constructor called
Derived's constructor called
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Q5:

```cpp
#include <iostream>
using namespace std;
class Box
{     double
width;

public:
```

```
    friend void printWidth(Box box);
void setWidth(double wid);
}; void Box::setWidth(double
wid)
{     width =
wid;
} void printWidth(Box
box)
{     box.width = box.width * 2;      cout <<
"Width of box : " << box.width << endl;
}
int main()
{
    Box box;
box.setWidth(10.0);
printWidth(box);
return 0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
Width of box : 20
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - The program defines a Box class with a private member width and a public member function setWidth() to set the width of the box. It also declares a friend function printWidth() to print the width of the box. In the main() function, it creates a Box object box and sets its width to 10.0 using setWidth().

Q6:

```
#include <iostream>
using namespace std;
class sample
{     int width,
height;

public:
    void set_values(int, int);
int area()
    {         return (width *
height);
    }
```

```
    friend sample duplicate(sample);
}; void sample::set_values(int a,
int b)
{     width =
a;     height
= b;
} sample duplicate(sample
rectparam)
{     sample rectres;     rectres.width =
rectparam.width * 2;     rectres.height =
rectparam.height * 2;     return
(rectres);
}
int main()
{     sample rect, rectb;
rect.set_values(2, 3);
rectb = duplicate(rect);
cout << rectb.area();
return 0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
24
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explenation - In the main() function, a sample object rect is created and its set_values() function is called to set its width to 2 and height to 3. Then, the duplicate() function is called with rect as an argument. Inside duplicate(), a new sample object rectres is created with width and height equal to double of rect's width and height, respectively. So, rectres will have a width of 4 and a height of 6. The area() function of rectres calculates the area, which is 4 * 6 = 24. This value is then returned to the main() function and printed using cout.

Q7:

```
#include <iostream>
using namespace std;
int Add(int X, int Y, int Z)
{     return X +
Y;
}
double Add(double X, double Y, double Z)
{     return X +
Y;
}
```

```
int main() {      cout <<
Add(10, 9);      cout <<
Add(4.5, 5.5);      return
0;
}
```

Error - The first error is that the function has three initialized variables and in function there only 2 variable get called. So, Here in int main() when Add() Called.Terminal can't get to know which Add() will be called.

Q8:

```
#include <iostream>
using namespace std;
void print(int i)
{     cout <<
i;
} void print(double
f)
{     cout <<
f;
} int
main(void)
{     print(5);
print(500.263);
return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
5500.263
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - Here two function with same name and different signature initialized and in int main() both function get called respectively.

Q9:

```
#include <iostream>
using namespace std;
class Overloading
{
```

```
private:
    int num;

public:
    Overloading()
    {         num = 5;
cout << num << "\n";
    }     void
operator++()
    {
        ++num;
    }     void
display()
    {         cout << num
<< endl;
    } };
int main()
{
    Overloading obj;
obj.display();
    ++obj;
    obj.display();
    ++obj;
obj.display();
return 0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
5
5
6
7
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - In This Code,++ Operator Overloading Initilized. 1st constructor gets called so output=5. 2nd Function gets called so output=5. Next ++ Operator Overloading Gets initialized so output=6. After Respectively ++ op Overloading initialized so output=7.

Q10:

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

```
class Distance
{
private:
    int feet;   // 0 to infinite
int inches; // 0 to 12 public:
    // required constructors
    Distance()
    {         feet
= 0;
inches = 0;
    }
    Distance(int f, int i)
    {         feet
= f;
inches = i;
    }     void
displayDistance()
    {         cout << "F: " << feet << " I:" << inches
<< endl;
    }
    Distance operator-()
    {         feet = -feet;
inches = -inches;         return
Distance(feet, inches);
    }     bool operator<(const
Distance &d)
    {         if (feet <
d.feet)
        {
return true;
        }         if (feet == d.feet && inches
< d.inches)
        {
return true;
        }
return false;
    } };
int main()
{
    Distance D1(11, 10), D2(5, 11);
if (D1 < D2)
    {         cout << "D1 is less than D2 "
<< endl;
    }
else
```

```
    {          cout << "D2
is less than D1 " <<
endl;
    }     return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
D2 is less than D1
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - The program defines a class Distance to represent distances in feet and inches, with methods to display distances and overloaded operators for negation and comparison. In the main() function, it creates two Distance objects and compares them using the < operator overload, which checks if one distance is less than the other based on feet and inches. The output indicates which distance is smaller.

Q11:

```
#include <iostream>
using namespace std;
class Complex
{
private:
    int x, y;
 public:
void input()
    {
        cout << " Input two complex number: " << endl;
cin >> x >> y;
    }
    Complex operator+(Complex obj)

    {
        Complex A;
        A.x = x + obj.x;
        A.y = y + obj.y;
return (A);
    }
    Complex operator-(Complex obj)
    {
```

```
        Complex A;
        A.x = x - obj.x;
        A.y = y - obj.y;
return (A);
    }
void disp()
     {         cout << x << " + " <<
y << "i"
             << "\n";
    }     void
print2()
    {         cout << x << " - " << y
<< "i"
             << "\n";
    } };
int main()

{
    Complex x1, y1, sum, sub;     x1.input();
y1.input();     sum = x1 + y1;     sub = x1 - y1;
cout << "\n Entered values are: \n";     cout << "
\t";     x1.disp();     cout << " \t";
y1.disp();     cout << "\nAddition of two complex
numbers: ";     sum.disp();     cout <<
"\nSubtraction of two complex numbers: ";
sub.print2();     return 0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
 Input two complex number:
 7  8
 Input two complex number:
4 6

 Entered values are:
       7 + 8i
       4 + 6i

Addition of two complex numbers: 11 + 14i

Subtraction of two complex numbers: 3 - 2i
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - This is the program for operator overloading concept. Here + and – operator overloading concept is used.The class complex is initialized in that two complex number taken from user and after it addition and subtraction done. And in main() functions get called.

Q12:

```cpp
#include <iostream>
using namespace std;
class Base1
{
public:
    char data;
};
class Base2
{
public:
    int data;
}; class Child : public Base1, public
Base2
{ public:
void show()
    {
        cout << Base2::data;
    } };
int main(void)
{
    Child d;
    d.show();
return 0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
0
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - This C++ code defines three classes: Base1, Base2, and Child. Child inherits from both Base1 and Base2. In Child, there's a method show() that outputs the data member of Base2. In the main function, an instance of Child is created and its show() method is called, which prints the data member of Base2.

Q13:

```cpp
#include <iostream>
using namespace std;
class Base
{     int
arr[5];
}; class Child1 : public
Base
{ }; class Child2 :
public Base
{ }; class GrandChild : public Child1, public
Child2
{ };
int main(void)
{
    cout << sizeof(GrandChild);
return 0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
40
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - In this program , This is Example of Multiple Inheritence Here one Base class is taken. It is inherited by two class child1 and child2 respectively. That two class also inherited by Grandchild. Here is Main(), sizeof(Grandchild) was returned, So, In base class arr with size 5 initialized. It's inherited by two child classes so 5+5=10 and size of int=4byte. So,Answer is 40.

Q14:

```cpp
#include <iostream>
using namespace
std; class A {
public:     void
display()
    {         cout << "
Inside A";
    } }; class B
: public A
{ public:
void display()
    {         cout << "
Inside B";
    } }; class C
: public B
{ }; int
main(void)
{
    C C;
    C.display();
return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
Inside B
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - Here Function Overriding is used. Here three class is initialized class a,b and c. class c inherit b and class b inherit a. In main() object of class c initialized. And display function called So firstly c inherit b so it's function called so answer is Inside b.

Q15:

```cpp
#include <iostream>
using namespace std;
class Base
{
private:
```

```
    int data1, data2;

public:
    Base(int a = 10, int b = 30) : data1(a), data2(b)
    {
    } }; class Derived :
public Base
{ public:
void show()
    {         cout << " data1 = " << data1 << " data2 = " <<
data2;
    } }; int
main(void) {
    Derived d;
d.show();
return 0;
}
```

Error - There are two Classes defined Derived and base. Base is parent class and Derived is child class. Derived class inherits Base class.But in Base class Data member is privately Declared.So,It can't accessible for child class.So,it give compile time error.

Q16:

```
#include <iostream>
#include <string> using
namespace std; class
Base
{ public:     virtual string
print() const
    {         return "This is Base
class";
    } };
class Child : public Base
{ public:     virtual string
print() const
    {         return "This is Child
class";
    }
};
```

```
void describe(Base p)
{     cout << p.print() <<
endl;
}
int main()
{
    Base b;
Child d;
describe(b);
describe(d);
return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
This is Base class
This is Base class
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```

Explanation - :In This code by using virtual key, This is example of Dynamic Polymorphysm. There are two class is called one is base and other is derived(child). Child class inherit base class.In both class print() function is virtually declared.In derived class one function named describe is declared. In that function's signature the object of base class is taken.In main(), Two object were taken one of base and other of child class. Then function describe two times called,First by class b object and other time is class child object. After compile The output will be from base class's print() function. Because object of child class object taken but in child class describe function has object of base class so output must be from base class.

Q17:

```
#include <iostream>
using namespace std;
class A { private:
    int id, salary;
    string name, designation;
 public:     void
getInfo()
    {         cout << "Enter id , name, designation and
salary:\n";         cin >> id >> name >> designation >>
salary;
    }     void
display()
```

```
    {         cout << "\n ====Employee Information=====\n";         cout <<
id << "\t" << name << "\t" << designation << "\t" << salary << endl;     }
public: }; class B { public:     A obj;     void disp()
    {
obj.getInfo();
obj.display();
    } };
int main() {
    B ob;
ob.disp();     A
obj;     return
0;
}
```

```
PS C:\ICT\SEM-2\ESFP-II> cd 'c:\ICT\SEM-2\ESFP-II\Practical-12\output'
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output> & .\'main.exe'
Enter id , name, designation and salary:
1001 manan CS 100

 ====Employee Information=====
1001    manan   CS      100
PS C:\ICT\SEM-2\ESFP-II\Practical-12\output>
```