What is operator overloading?

Ans: In C++, Operator overloading is a compile-time polymorphism. It is an idea of giving special meaning to an existing operator in C++ without changing its original meaning.

Example:

```cpp
#include <bits/stdc++.h>
using namespace std;
class sum{
    int num,res;
    public:
    void input(){
        cin>>num;
    }
    sum operator+(sum &ob){
        sum res;
        res.num=this->num+ob.num;
        return res;
    }
    void output(){
        cout<<num;
    }

};
int main(){
    sum ob1,ob2,ob3;
    ob1.input();
    ob2.input();
    ob3=ob1+ob2;
    ob3.output();

    return 0;
}
```

Output:

```
/home/nahid/Generic/assignment-test
5
5
10
Process returned 0 (0x0)   execution time : 3.053 s
Press ENTER to continue.
```

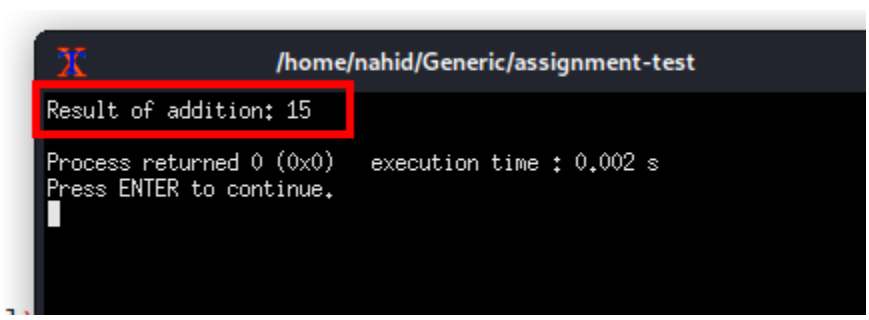
What is friend operator overloading? Explain with example

Ans: Friend operator overloading is a feature in C++ that allows you to declare friend functions or classes within a class definition. This provides these friends with unrestricted access to the class's members.

Example:

```cpp
#include <bits/stdc++.h>
using namespace std;
class Sum {
private:
    int value;
public:
    Sum() : value(0) {}
    Sum(int val) : value(val) {}
    int getValue(){
        return value;
    }
    friend Sum operator+(Sum& obj1, Sum& obj2);
};
Sum operator+(Sum& obj1, Sum& obj2) {
    Sum result;
    result.value = obj1.value + obj2.value;
    return result;
}
int main() {
    Sum obj1(5);
    Sum obj2(10);
    Sum result = obj1 + obj2;
    cout << "Result of addition: " << result.getValue() << std::endl;
    return 0; }
```

Output:



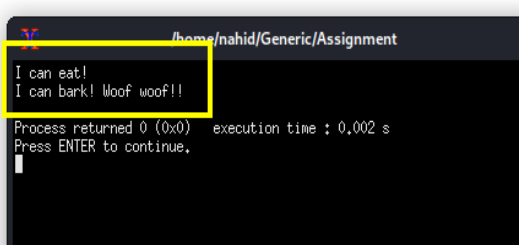
What is Inheritance?

Ans: Inheritance is one of the key features of Object-oriented programming in C++. It allows us to create a new class (derived class) from an existing class (base class).

Example:

```cpp
#include <iostream>
using namespace std;
class Animal{
public:
    void eat()
    {
        cout << "I can eat!" << endl;
    }
};
class Dog : public Animal{
public:
    void bark()
    {
        cout << "I can bark! Woof woof!!" << endl;
    }
};
int main(){
    Dog dog1;
    dog1.eat();
    dog1.bark();
    return 0;
}
```

Output:



Inherit Protected Data:

```cpp
#include <bits/stdc++.h>
using namespace std;
class Vehicle{
 protected:
     string model;
     int year;
};
class Property{
 protected:
     string color;
     double weight;
};
class car: public Vehicle, public Property{
public:
     void input(){
     cout<<"Enter Car Model Year color weight: ";
      cin>>model>>year>>color>>weight;
     }
    void show(){
    cout<<"Car Info:"<<endl;
    cout<<model<<" "<<year<<" "<<color<<" "<<weight<<endl;
    }
};
int main(){
    car ob1;
    ob1.input();
    ob1.show();
return 0;
}
```

Output:

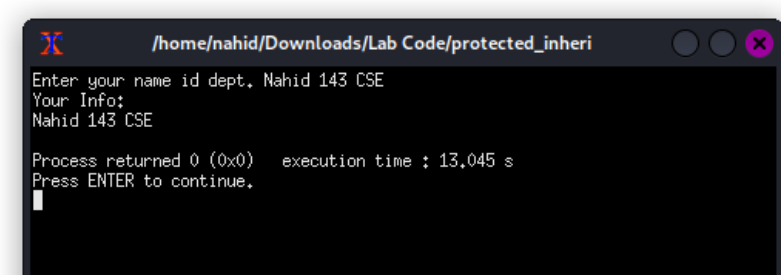
```
/home/nahid/Downloads/Lab Code/protected_inheri
Enter Car Model Year color weight: Toyota 2024 Red 250
Car Info:
Toyota 2024 Red 250

Process returned 0 (0x0)   execution time : 20.500 s
Press ENTER to continue.
```

Constructor, Destructor & Multiple Inheritance:

```
#include <bits/stdc++.h>
using namespace std;
class Student{
 protected:
     string name;
     int id;
};
class Info{
 protected:
     string dept;
};
class Person: public Student, public Info{
public:
     Person(){
     cout<<"Enter your name id dept. ";
      cin>>name>>id>>dept;
     }
    ~Person(){
    cout<<"Your Info:"<<endl;
    cout<<name<<" "<<id<<" "<<dept<<endl;
    }
};
int main(){
    Person ob;
   return 0;
}
```

Output:



What is Generic Function?

Ans: Generics is the idea to allow type (Integer, String, … etc and user-defined types) to be a parameter to methods, classes and interfaces. For example, classes like an array, map, etc, which can be used using generics very efficiently.

Example:

```
#include <bits/stdc++.h>
using namespace std;
template <typename N>
N add(N a, N b)
{
    return a+b;
}
int main()
{
    cout<<add<int>(5,5)<<endl;
    cout<<add<double>(5.5,15.7)<<endl;
    return  0;
}
```

Output:

```
/home/nahid/Downloads/Lab Code/gen-func
10
21.2

Process returned 0 (0x0)   execution time : 0.003 s
Press ENTER to continue.
```

What is Generic Class?

Ans: Generic class is a class that is designed to work with different data types while providing a common interface and implementation

Example:

```
#include <iostream>
using namespace std;
template <class N>
class Nahid{
    private:
    N a,b;
    public:
    Nahid(N x,N y){
        a=x;
        b=y;
    }
    void area(){
        cout<<"A*B ="<<a*b<<endl;
    }
};
int main(){
    Nahid <int> ob(3, 3);
    Nahid <double> ob1(3.3, 3.9);
    ob.area();
    ob1.area();
}
```

```
/home/nahid/Generic/gen_cls3
A*B =9
A*B =12.87

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

What is exception Handling?

Ans: Exception handling is a mechanism in programming languages that allows a program to deal with unexpected or exceptional situations during runtime.

Example:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    while (1)
    {
        int a,b;
        cout<<"Enter value of a & b: ";
            cin>>a>>b;
        try
        {
            if (b==0)
            {
                throw 1;
            }

            cout<<"Result = "<<a/b<<endl;

        }
        catch (...)
        {
            cout<<"Cannot divided by Zero/One"<<endl;
        }

    }
    return 0;
}
```

Output:

```
/home/nahid/Generic/execp-hand
Enter value of a & b: 10 2
Result = 5
Enter value of a & b: 20 5
Result = 4
Enter value of a & b: 10 0
Cannot divided by Zero/One
Enter value of a & b:
```

What is virtual function?

Ans: In object-oriented programming, a virtual function is a member function of a class that can be overridden in a derived class. The concept is a key feature of polymorphism, allowing a base class pointer or reference to invoke a function that is defined in a derived class.

Example:

```cpp
#include <iostream>
using namespace std;
class Base {
   public:
    virtual void print() {
        cout << "Base Class Function" << endl;
    }
};

class Derived : public Base {
   public:
    void print() {
        cout << "This is Derived class function-Nahid" << endl;
    }
};

int main() {
    Derived derived1;

    // pointer of Base type that points to derived1
    Base* base1 = &derived1;

    // calls member function of Derived class
    base1->print();
    return 0;
}
```

Output:

```
/home/nahid/Downloads/Lab Code/virtual_functiom
This is Derived class function-Nahid

Process returned 0 (0x0)   execution time : 0.004 s
Press ENTER to continue.
```

What is Pure Virtual Function? And Abstraction

Ans: A pure virtual function in C++ is a virtual function that is declared in a base class but has no implementation there. It is marked with the "pure specifier" = 0.A class containing at least one pure virtual function isconsidered an abstract class.

Example:

```cpp
#include<iostream>
using namespace std;
class Flower{
public:
    virtual void showitem()=0;
};
class Rose : virtual public Flower{
public:
    virtual void showitem()
    {
        cout<<"Sell Rose"<<endl;
    }
};
class Marigold : virtual public Flower{
public:
    virtual void showitem()
    {
        cout<<"Sell Marigold"<<endl;
    }
};
int main(){
    Flower *r=new Rose();
    r->showitem();
    Flower *m=new Marigold();
    m->showitem();
    return 0; }
```

Output:

```
/home/nahid/Downloads/Lab Code/pure_virtual_function
Sell Rose
Sell Marigold

Process returned 0 (0x0)   execution time : 0.004 s
Press ENTER to continue.
```