

1- Write a definition for each of the following terms:

- **Encapsulation**
It's a fundamental of Object Oriented Programming concept that combine data and method in one unit called Object. It help developer to hide data into the object.
- **Data Hiding**
This an Object Oriented Programming functionality that keep data secure from using outside the current object. That no one can access member data directly, but above an interface called member functions.
- **Polymorphism**
As it name, Poly-Morph, it mean that the one member function can be used in many shapes by overloading. Ex: when we define a default constructor with no params, and then define another constructor with the same name with another parameters. This is the Polymorphism.
- **Overloading**
As we mentioned above, Overloading means that we can use the same function name to define another function with another functionality, parameters and return type. Ex: consider we have a function called **int sum(int a, int b)**; this function will sum integers but if we want it to sum double, **double sum(double a, double b)**; this is overloading.
- **Object**
In OOP **object** is the base unit of it, Object is everything that we can make model for it, Ex: Car, Person, Employee, Bee, Laptop, etc.
- **Class**
Class is a blue print of an **Object**, this blue print contain common functions of this object, but can't store objects data.
- **Member Function**
This is a function that can be accessed from the current object, this function is an interface between object member data and the outer world (user/developer).
- **Constructor**
This is a special member function that will run automatically when an object is created.
This function used for initializing member data.

- Destructor

This is also a special member function that will run automatically at the end of the object **Life-Cycle**. This function used to de-allocate or free memory from this object.

- Friend Function

It is a function that can access friend class member data, this isn't a member function. But it is a function as we study in structural programming.

- const Member Function

This is a const function that usually display data, It can't update or edit object member data as it const.

- const Object

This make object member data const like as it created we can't change any of it's Member data.

- Static Class Data

This data is related to the class not any object, as it will be shared between all objects taken from this class. Ex: Consider we have a class Student; and I want to count how many students objects created from this class. What should you do? You may create list and push all objects on it, then count the list. Yes, this is a bad-practice. So, we have another way, we will create a static member data from the class Student with name **static int _counter**; Edit constructor, **Student::_counter++**; then edit destructor, **Student::_counter--**;

And then when your want to count the students do this: **Student::_counter**;

*** Note Make sure that you have access to _counter, as it private data, you can create an interface function to access it's value.**

- Operator overloading

This is like a function overloading, but in operator. Ex: Determine we have a class **Rectangle**; with data: **int width, height**; Then create two objects **R1, R2**;

What should you do if you want to sum those rectangles? You know?!

We will make + operator overloading like this: **Rectangle operator+(Rectangle R2)**; then return new object with the sum of **this** and **R2**;

2- Demonstrate the different ways for initializing a constructor.

There are two ways initializing constructor

Default and Parametrized Constructor

3- Demonstrate the operators that can be overloaded.

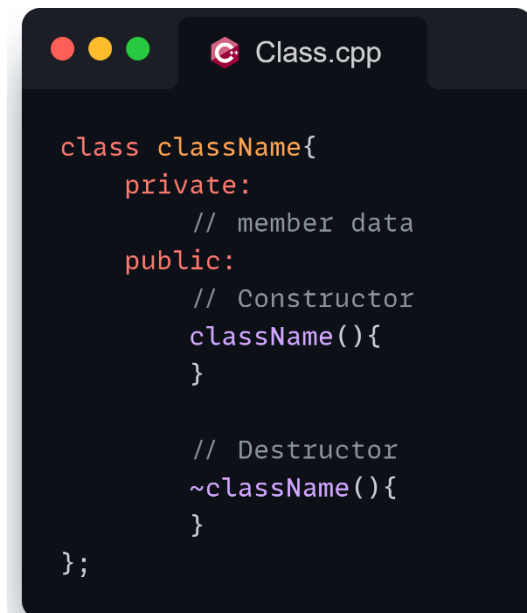
+	-	++(pre)	--(pre)	(post)++	(post)--
/	*	%	==	!=	>
>=	<=	+=	-=	*=	/=

4- Demonstrate the operators that cannot be overloaded.

Sizeof, typeid, Scope resolution operator(::), dot operator(.)

5- Write down a syntax for each of the following:

- class



```
class className{
    private:
        // member data
    public:
        // Constructor
        className(){
        }

        // Destructor
        ~className(){
        }
};
```

- Constructor
We created it above.
- Destructor
Also

- definitions for a member function inside and outside the class



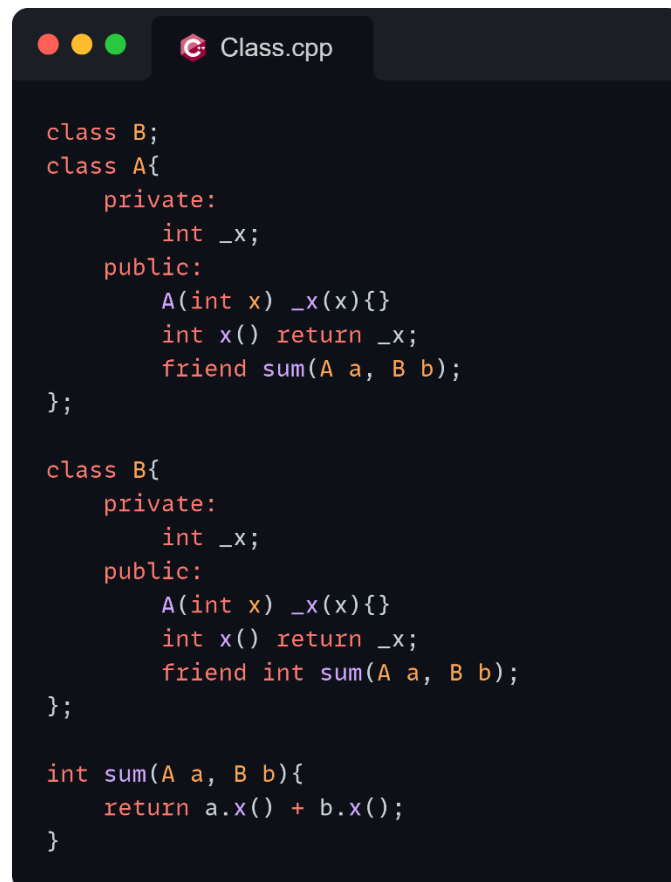
```
Class.cpp

class className{
public:
    // Inside-Class
    className(){
    }
};

// Out-Class
className::className(){
}
```

- friend function

-



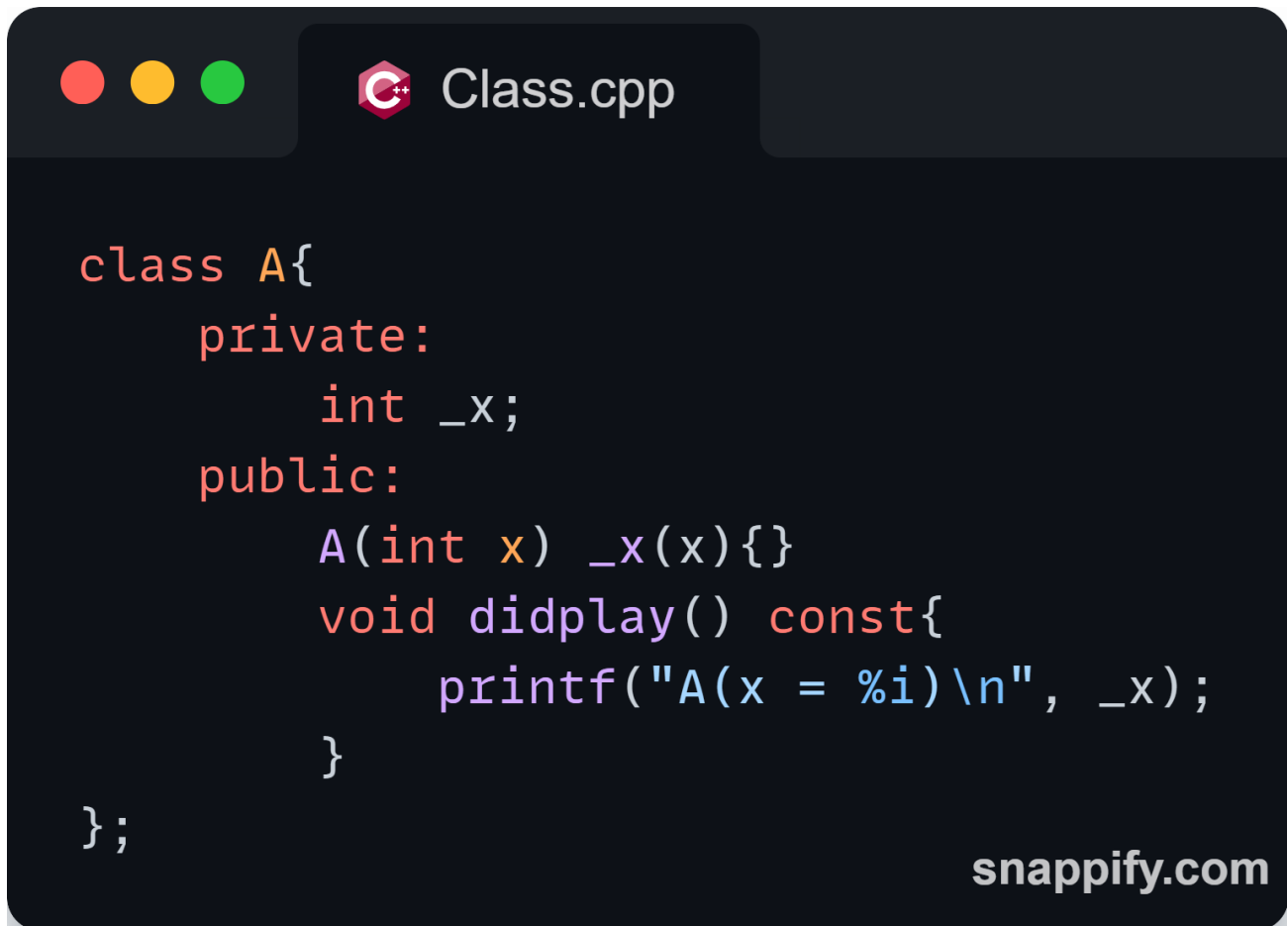
```
Class.cpp

class B;
class A{
private:
    int _x;
public:
    A(int x) _x(x){}
    int x() return _x;
    friend sum(A a, B b);
};

class B{
private:
    int _x;
public:
    A(int x) _x(x){}
    int x() return _x;
    friend int sum(A a, B b);
};

int sum(A a, B b){
    return a.x() + b.x();
}
```


- const function



```
class A{
    private:
        int _x;
    public:
        A(int x) _x(x){}
        void didplay() const{
            printf("A(x = %i)\n", _x);
        }
};
```

snappify.com

Overloading an Operator.



```
class A{
    private:
        int _x;
    public:
        A(int x) _x(x){}
        void display() const{
            printf("A(x = %i)\n", _x);
        }
        A operator++(){
            return A(++_x);
        }
};
```

snappify.com