C++ Assingment 2

1. What are the benefits and drawbacks of operator overloading?

**Benefits of function overloading are as follows:**

1. The main advantage of function overloading is that it improves code readability and allows code reusability.
2. The use of function overloading is to save memory space, consistency, and readability.
3. It speeds up the execution of the program
4. Code maintenance also becomes easy
5. Function overloading brings flexibility to code

**Drawbacks of function overloading are as follows:**

1. Member function declarations with the same name and the same parameter types cannot be overloaded if any of them is a static member function declaration.
2. The main disadvantage is that it requires the compiler to perform name mangling on the function name to include information about the argument types.

2. Can you overload the assignment operator (=) in C++? If so, how would you ensure proper behavior?

Yes, we can overload the assignment operator,”=”, is the operator used for Assignment. It copies the right value into the left value. Assignment Operators are predefined to operate only on built-in Data types. Assignment operator overloading is binary operator overloading. Overloading assignment operator in C++ copies all values of one object to another object. Only a non-static member function should be used to overload the assignment operator.

Example :

class C

{};

int main() {

C c1,c2;

c1 = c2;

return 0;

}

3. Explain the difference between member function and non-member (friend) function overloading for operators ?

Friend Function -> It is basically a function that is used to access all private and protected members of classes. It is considered as a non-member function of class and is declared by the class that is granting access.

class A{

private:

{

public:

{

friend void check();

}

void check();

}

}

**Member Function:** It is basically a function that can be declared as members of a class. It is usually declared inside the class definition and works on data members of the same class. It can have access to private, public, and protected data members of the same class.

class A{

private:

{

public:

{

void check();

}

A::void check();}

}

4. Design a class Vector2D and overload the arithmetic operators (+, -, \*, /) for vector addition, subtraction, scalar multiplication, and division (by a scalar).

Is it possible to overload the comparison operators (==, !=, <, >, <=, >=) for custom classes? If so, what considerations should be taken into account?

Can you overload the stream insertion (<<) and extraction (>>) operators for your Vector2D class to allow easy printing and reading from streams?

Describe a scenario where overloading the logical operators (&&, ||, !) for a custom class might be useful.

Discuss the potential ambiguity that could arise when overloading the subscript operator ([]) for a class. How can this ambiguity be resolved?

Can operator overloading be used to implement the concept of immutability (unchanging state) for a class? Explain your answer.

When overloading operators, what are some best practices to ensure code clarity and maintainability?

#include <iostream>

class A{

public:

int x;

int y;

public :

A(int a,int b){

x = a;

y=b;

}

A operator +(A a){

A m (x + a.x, y + a.y);

return m;

}

A operator -(A a){

A m (x - a.x, y - a.y);

return m;

}

A operator \*(A a){

A m (x \* a.x, y \* a.y);

return m;

}

};

int main()

{

A a1(5,6);

A a2(6,3);

A a3 = a1 + a2;

std::cout << "Addition is : "<< a3.x << " + "<< a3.y << std::endl;

A a4 = a1 -a2;

std::cout << "Substraction is : "<<a4.x << "-"<< a4.y<< std::endl;

A a5 = a1 \* a2;

std::cout << "Multiplication is : "<<a5.x <<"\*"<<a5.y<< std::endl;

return 0;

}

Explaining Code :

* I create a class name A. Here we have to do some task like (add, sub, multiply and division).
* With the operator like(+, -, \*, /) we have to perform an operator overloading to perform the task.
* Now in this we have to add two objects and put this value in the other object in same class.
* This is what operator overloading so we make some function name operator for any one operator we have to perform the task.

5. What is the core concept behind function overloading?

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. C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading. For example, we can overload an operator ‘+’ in a class like String so that we can concatenate two strings by just using +. Other example classes where arithmetic operators may be overloaded are Complex Numbers, Fractional Numbers, Big integers, etc.

// Example

class A {

statements;

};

int main()

{

A a1, a2, a3;

a3 = a1 + a2;

return 0;

}

6. How does the compiler differentiate between overloaded functions with the same name?

Ans : The compiler differentiate between the functions based on the arrangements of the arguments or other parameter. So we can have multiple functions with the same name but parameters and function overloading will resolve at build time based on the function.

7.