Assignment No - 15

Q1] What is concept of Function overloading

→ - Function overloading is a programming concept that allows multiple functions in the same scope to have the same name but different parameter list.

- The compiler distinguishes these functions based on the number, type or order of their parameters.
- It is a form of polymorphism in object-oriented programming, enabling functions to perform different tasks depending on their arguments.
- Function overloading is resolved at compile time so it is a type of compile time polymorphism.
- It improves code readability and reusability.
- Functions must have the same name but differ in its number of parameters, Types of parameter and order of parameters.

Overloading.C++

```
#include <iostream>
using namespace std;
class Arithmatic
{ public :
    int Addition (int no1, int no2)
    { int Ans = 0;
      int Result = no1 + no2;
      return Result;
    }
    int Addition (int no1, int no2, int no3)
    { int Ans = 0;
      int Result = no1 + no2 + no3;
      return Result;
```

```
float Addition (float no1, float no2)
{ float Ans = 0;
  float Result = no1 + no2;
  return Result;
}
float Addition (float no1, float no2, float no3)
{ float Ans = 0;
  float Result = no1 + no2 + no3;
  return Result;
}
};
int main
{ Arithmatic obj1;
  int ans = 0;
  ans = obj1. Addition (10,20);
  Cout << "10 + 20 = " << ans << "\n";
  ans = obj1. Addition (10,20,30);
  Cout << "10 + 20 + 30 =" << ans << "\n";
  ans = obj1. Addition (10.5f, 20.5f);
  Cout << "10.5f + 20.5f =" << ans << "\n";

return 0;
}
```

Q2] Explain difference between constant function and non-constant function.

→ Const function:

A constant function is a member function of a class that cannot modify any of the member variables of the class.

It ensures the integrity of the object's

state when the function is called.
- Declaration - use the const keyword after the function declaration.
- It can call other constant functions.

Non-constant Function
A non-constant function is a regular member function of a class that can modify the member variables of the object.
- It can modify member variables.
- Can call both const and non const functions.

Q3] Explain the concept of member initialization list of constructor in C++.

→ In C++, the member initialization list is a feature that allows you to initialize class member variables directly when a constructor is called, rather than assigning values to them the constructor body.
It is more efficient, especially for constant, reference or non-default constructible members.

- Syntax -
The member initialization list is placed after the constructor's parameter list and a colon (:). Each member is initialized with its value in parentheses

e.g.
```
class className {
    private :
        int x;
        int y;
    public :
```

```
    ClassName (int a, int b): x(a), y(b)
    {  //
    }
};
```

Q4) How can we initialize the constant characteristics of class?

→ In C++, constant data members of class must be initialized using the member initialization list in the constructor. This is because const variables cannot be modified after they are initialized. and the initialization list provides the only way to assign them a value during the construction of an object.

eg. Constcharacteristics.cpp

```
#include<iostream>
using namespace std;
class Demo
{ public:
    int i;
    const int j;
    const int k;
    Demo (int a, int b, int c): j(b), k(c)
    { i = a;
    }

int main()
{ Demo obj1 (10, 20, 30);
  cout << "obj1.i :: " << obj1.i << endl;
  cout << "obj1.j :: " << obj1.j << endl;
  cout << "obj1.k :: " << obj1.k << endl;
  return 0;
}
```

Q6] Explain what are the limitations of static function of class?

→ 1] Static function cannot directly access non-static members of the class, such as member variables or non static member functions.
2] Static functions do not have a this pointer because they belong to the class, not to an object.
3] Static functions can only acces to other static member functions, and static variables of the class.
They cannot directly work with instance specific data unless it is explicitly passed to them as arguments.
4] Static functions cannot be virtual, so they do not support runtime polymorphism or function overriding
This is because polymorphism works through a vTable, which relies on this pointer.
5] Static functions cannot be qualified as const or volatile because these qualifiers are associated with the state of an object, which static fonctions do.

Q7 How to initialize the static constant characteristics of a class.

→ We can initialize the static const members of a class either inside the class declaration or outside the class definition

1] Inside the class -
If the static const member is of an integral or enumeration type and you want to initialize it at compile time, you can do it directly in class declaration.
eg.

```
class MyClass
{ public :
    Static Const int value = 10;
};
```

2] Outside the class
If we static const member is not initialized inside the class then we must initialized it outside the class or after the class definition.
e.g.

```
#include <iostream>
class MyClass
{ public :
    Static const int value; // Declaration Only
};
const int MyClass :: value = 10; // Definition outside class

int main()
{ Cout << MyClass :: value << endl;
return 0;
}
```

Q8] Explain why constant object can ~~not~~ call constant member functions of class?

→ A constant object is restricted to only call const member functions because
- A const object guarantees that ift will not be modified.
- A const member function does not allow modification of the object's state, thus it is safe to call from const object.
- A non const member function could modify the object which which violates the const promise of the object.
- when we declare an object as const the compiler ensures that no operations that could modify the object are allowed. If we try to call a ~~Ca~~ non-const member fun? the compiler will raise an error.

Q9] Can we call static member function of a class using object of class.

→ Yes, Technically we can call a static member function of a class using an object of that class, but it is generally considered bad practice as static functions are meant to be accessed directly through the class name and not through an object instance.
As they don't operate on specific object ~~class~~ data.
Using an object to call a static function can be misleading.

The recommended way to call a static function is using the class name followed by the scope resolution operator (::)

Q10] What is difference between constant input argument and non const arguments of function with program.

→ Constant Input Argument

- A const argument is a function parameter that cannot be modified within the function.
- It is typically used to prevent accidental changes to the input and to signal to the caller that the argument will remain unchanged.
- The function can read but not modify the value of const argument.
- Protect the input data from unintended modification.

Non Constant Arguments:

- It is regular parameter that can be modified within the function.
- The function is free to change the value of the parameter, but changes will not affect the original argument unless passed by reference.
- The argument can be freely modified inside the function.

eg.

```
#include <iostream>
Using namespace std;
// function with const argument
void ConstArgumentfunction (const int x)
{ cout << " const argument : " << x << endl;
}
// function with non const argument
void nonConstArgument (int x)
{   x = x+1 ;
    cout << " non const argument : " << x << endl;
}
int main()
{   int num = 10;
    ConstArgumentfunction (num);    // 10
 nonConstArgument (num);            // 11

  cout << " original value : " << num << endl;
```