PAGE No
DATE / /

Assignment -15

Q1] What is use of Function overloading?

Ans

1] ther term Polymorphism in object orientation is defined as single name & multiple behaviours

2] In C++ & Java polymorphism is divided into two types

1] Compile - time polymorphism

2] Run - time polymorphism

3] To achieve compile time polymorphism we use the term called overloading

There are two types of overloading

1] Function overloading

2] operator overloading.

4] According to the creator of C++, polymorphism is considered as syntatic sugar which just adds a flavour in our application.

5] In case case of function overloading we can define multiple functions in a class with the same name & with different prototype.

~~Example~~ :

6] Due to concept of function overloading as a user of class, there is no need to remember names of different functions.

```
class Demo
{
    public :
        int addition (int inum1, int inum2)   // Additional
        {
            int ans = 0;
            ans
```

16] But when when we compile the code, compiler changes ~~the~~ the name of every function with the mangled-name (edited name)

Q5] Can we use the concept of this pointer in case of static member function of class?

Ans:] No, the concept of this pointer can only be used in case of constructor, non-static function, destructor.

·] This because the first hidden implicit parameter of ~~function~~ caller object ~~has~~ has address of the object.

·] This address is stored inside this pointer, which ~~#~~ is the first implicit argument of function.

·] But incase of static member function, as they are created irrespective of object, this pointer cannot be used in their case.

Q6] Can we call static member function of a class using object of a class?

Ans:] Yes, we can call static member function of ~~#~~ a class using object of a class ~~but~~

·] But the compiler internally converts it into name of class.

PAGE No
DATE / /

- Example

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

class Demo
{
    public :
        int iNum1 = 0 , iNum2 = 0;
        Static int ~~=~~ iNum3;

    Demo (int iValue1 = 10 , int iValue2 = 20)
    {
        iNum1 = iValue1;
        iNum2 = iValue2;
    }
    Static void fun()            // Static member function
    {
    cout<<"Inside Static method function"<<endl;
     cout<< "Value of iNum3 is ~~[illegible]~~" << iNum3<<endl;
    }
};

int Demo :: iNum3 = 35;

int main()
{
    Demo obj1(20,30);

    Obj1.fun();   // Accessing Static member function
    return 0;          using object of class
}
```

OUTPUT

```
cmd
Inside static method
function
value of iNum3 is 35
```

Q7] What are the ~~t~~ limitations of static function of a class?

Ans

1] Static function can access static characteristics of class only.

Q8] How to initialise static constant characteristics of a class?

Ans I] ~~Initialise the~~

```
class Demo {
    public:
        static const int m;
    };
const int Demo :: m= 11;
```

Q9] Why constant object can ~~cat~~ not call constent member functions of a class?

Ans

1] Constant object is such a object whose all the characteristics are considered as considered.

2]

Q2] What is difference between constant function & non - constant function in C++ ?

Ans Constant function is such a function which cannot change values of characteristics inside its body.

e.g.

```
class Demo{
        public:
          int i,j,k;

          void change() const
            {
                i++;        //NA
                j++;        //NA
                k++;        //NA
            }
};
```

Non Constant is such a function ~~whose~~ in ~~ch~~ which characteristics of a class are modifiable.

```
class Demo {
        public:
          int i,j,k;
          void change()
          {
               i++;      // Allowed
               j++;      // ~~A A~~ Allowed.
          }
```

Q3] What is meant by member initialisation list of Constructor in C++?

Ans 1] In a class we can immediatetely initialise the constant characteristics, but it is a bad programming practice.

2] If we initialise const characteristics inside constructor we will get error generated by the compiler because the constructor gets called after allocating the memory for the object.

3] After allocating memory, the constant characteristic may contain some default values & we are trying to initialise that values again.

4] To avoid this we use the concept of member initialization list, by using which we initialise the values before entering the constructor

5] When we call object memory gets allocated for characteristics through constructor, but using member initialisation list value gets ~~all~~ initialised before entering the constructor.

Example

```
class Demo{
        int i~~,j,k~~;
        const int j;
        const int K;                                      j=b    k=c
            Demo( int a, int b, int c): j(b), k(c)
            {  i=a;
            }
```

PAGE No
DATE / /

Q4] How to initialise constant characteristics of a class?

Ans
1) The constant characteristics can be initialsied immediately, but it is a bad programming practice
2) There member initialisation list must be used inorder to initialise constant characteristics of class, by using which we initialise values before entering the constructor.

~~Q5]~~ Example
```
class Demo{
        int i;
        const int j;
        const int k;
        Demo (int a, int b, int c) : j(b), k(c)
        {
          i=a;
        }
```

Q16] What is the difference between constant input arguments and non constant input argument?

Ans 1) Constant input arguments : If the function argume are constant we cannot change its value inside function

Example
```
void display (int i, const int j)
{
 }
```

Inside this function value of j is non-modifi

2] <u>Non-constant input arguments</u>:

If the function arguments are non-constant they are modifiable inside the function. i.e its value can be changed.