Assignment No:- 14

Q1] What do you mean by class variable and instance variable of class.

→ Class Variable.
A class variable is a variable that is shared by all instances of a class.
It is declared using the static keyword and belongs to the class itself. rather than any specific object.
- It only one copy of the variable exists for the entire class.
- Can be accessed using the class name or through any instance.
- Exists for the entire lifetime of the program
- it stored in the static memory segment.

Instance of class.
It refers to a specific object created from a class blueprint. Each instance has its own copy of instance variables and can use the class's method's.
- Each instance has its own memory for instance variables.
- They can access non-static members of class.
- Instances are dynamically or automatically created.

Q2] What is mean by argument? Explain the term Default argument

→ An argument refers to the actual value or data that you pass to a function or method when you call it.

Arguments are used to provide input so that the function can perform its intended task.
- In argument the actual values or expressions provided during the function call.
- There are two types of arguments in the functions 1) Regular/Compulsory argument.
2) Default Argument.

Default Argument -
It is type of argument which is consider as a optional, if we skip that parameter while calling function then its default value get consider.
- In default argument all default argument must be at last of functions argument list. Otherwise Compiler will generate error.

Q3) Difference between static and non-static characteristics of a class.

→

	Static characteristics	Non-Static characteristics.
1)	Static char can access using class name.	It can access using instance of class.
2)	It can accessed by static and non-static methods.	It can't accessed inside a static method.
3)	It reduced the amount of memory used by a program.	It do not reduce the amount of memory used by a program.

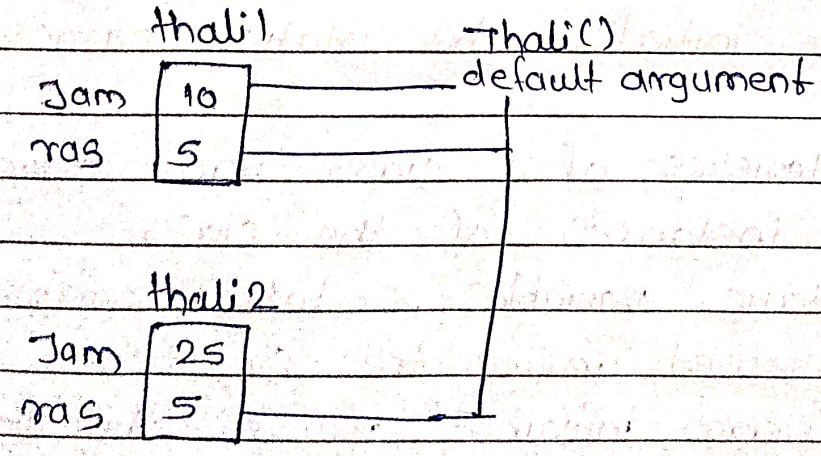
4) It allocated only once, at the time of class loading.	It allocated each time an instance of the class is created.
5) It can be access from any part of the program	It can be access only within the class or its instance.
6) It exists for the entire lifetime of program	It exists for the life-time of object.
7) It is like a global variable and is available to all methods	It is like a local variable and they can be access through only instance of a class.

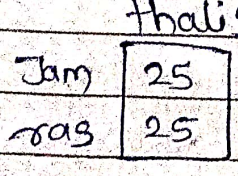
Q4) Explain the term parameterized constructors with default arguments.

→ A parameterized constructor in C++ is a constructor that accepts one or more parameters to initialize an object.
When default values are provided to one or more of these parameters, it becomes a parameterized constructor with default arguments.
eg.

```
#include <iostream>
using namespace std;
class Thali
{ public:
    int Jamun;
    int Rasgulla;
```

```
// Parameterized constructor with default arguments.
Thali (int jam = 10, ras = 5)
{  jamun = jam;
   rasgulla = ras;
}
void display()
{  cout << "Jamun " << jamun << endl;
   cout << "rasgulla " << rasgulla << endl;
   cout << " ----- " << endl;
}
};
int main()
{  Thali thali1;
   thali1. display();
   Thali thali2;
   thali2. display(25);
   Thali thali3;
   thali3. display (25,25);
}
```





Q5] What is the concept of name mangling? Explain in detail.

→ When we compiled the code then compiler change the name of every function with mangled name.
When we overload function called name of function is same due to the concept of name mangling the compiler change the name with new name.
According to above point we conclude that there is no such concept of function overloading after the program get compiled.

Name Mangling

int Addition (int no1, int no2)

Addition@ii = Addition @ ii

ii → initials of datatypes of every parameters.

Q6] How do we initialize the static characteristics of a class.

→ Static characteristics of a class are shared among all instances of the class
Unlike non-static variables, static variables are not initialized inside the constructor.
Instead they are initialized outside the class definition.
To initialize static characteristics
1] Declare the static variable inside the class.
2] Define and initialize the static variable outside the class.

e.g.

```
#include <iostream>
using namespace std;
class Thali
{ public :
    int jamun;
    int rasgulla;
    static int lonche;
  // parameteerized constructor with default arg
  Thali (int jam = 10, int ras =5)
  { jamun = jam;
    rasgulla = ras;
  }
  void display()
  { cout << "jamun =" << jamun << endl;
    cout << "rasgulla =" << rasgulla << endl;
    cout << "- --- " << endl;
  }
};
int Thali :: lonche = 111;  //assign of static variable.
int main ()
{ // access using classname :: staticvariablename.
  cout << "value of lonche : " << Thali :: lonche << endl;

  Thali thali1;
  thali1. display();
  cout << "size of any object => " << size of (thali1)
                                   << endl;
  cout << "value of lonche using object :" << thali1. lonche
                                   << endl;

  Thali thali2(25);
  thali2. display();
return 0;
}
```

Q7] Can we access private non static characteristics of a class from static method? Explain with example.

→ No,
Static methods cannot directly access non-static members of a class, including private ones, because static methods are not tied to any specific object of the class.
They belong to the class itself, not to any instance, and thus do not have access to instance-specific data.

Q8] Is it possible to create private static characteristics of class? explain with example.

→ Yes, it is possible to create private static characteristics in a class.
In C++, static members can have any access specifier, including private.
This allow the class to control access to static characteristics while still sharing them among all objects of the class.

e.g.

```
#include <iostream>
using namespace std;
class Example {
  private:
    static int count;
  public:
    static void incrementCount() {
        count++;
    }
```

```
    static void displayCount() {
        cout << "Count: " << count << endl;
    }
};
int Example :: count = 0;
int main()
{ Example :: incrementCount();
  Example :: incrementCount();
  Example :: displayCount();
return 0;
}
```

Q9] What is the lifetime (scope) of static characteristics of class.

→ The lifetime of static characteristics in a class refers the to the duration for which these characteristics exist in memory.
In C++. the scope of static characteristics is tied to the entire program execution rather than any individual object.

Q10] What is mean by static behaviours! explain with example.

In the context of class, static behaviours refer to static methods that belong to the class itself rather than any specific instance of the class.
These methods can be called without creating an object of the class.

e.g.

```
#include <iostream>
using namespace std;
class Calculator
{ public:
    static int add(int a, int b)
    { return a+b;
    }
    static int multiply(int a, int b)
    { return a*b;
    }
};
int main()
{ cout<< "Addition :" << Calculator :: add (5,10) << endl
  cout<< "Multiplication :" << Calculator :: Multiply
                              (4,6) << endl;
return 0;
}
```