Q1] What do you mean by class variable & instance of variable of class?

Class variable are the static characteristics of a class
Instance variable are the non-static characteristics of class

1> Class Variable

It is associated with a class rather than individual object.
Memory for static characteristics gets deallocated by default & only once.
These should be initialised outside class using 'Scope Resolution Operator (::)'.
We can access them without creating the object with the help of class name & scope resolution operator.

2> Istance variable

It belongs to a specific object (instance) of the class.
Memory for non-static characteristics get allocated separately for each object.
These are accessed using a specific object of a class.
These are used to store data unique to each instance of a class.

Q2] What is mean by Argument ? Explain Default Argument.

There are two types of Arguments :-

1> Regular / Compulsory Argument
2> Default Argument

Argument refers to the value or variable passed to a function.

• Default Argument

It is a type of Argument which is considered as a optional, if we skip that parameter while calling the function then its default value gets considered.

If we are using the concept of Default Argument then we have to follow one rule that all the default Argument must be at the last of function Arguments, list otherwise the compiler will generate an error.

Program :- default.cpp

03] Difference between Static & Non-static characteristics of a class

Points	Static	Non-static
Association	Allocated once, shared across all the objects	Belongs to individual objects (instances)
Storage	Belongs to class itself	Separate storage for each object
Initialization	Outside the class for static data	Done per object typically through conductor
Access	Accessed using the class name or object	Accessed only via an object
Member functions	Cannot access non static members	Can access both static & non-static member
Use Case	Shared or global state. utility functions	Instance-specific data & behaviour.

Q4] Explain parametrized Constructor with Default Arguments

We can use the concept of default Arguments incase of constructor.

If we create a parametrized constructor which uses default arguments in it then that type of constructor is called, 'Parametrized Constructor with Arguments

It allows us to pass specific values to initialise an object while also providing default values if no arguments ~~for~~ are supplied during object creation.

Ex Program:- Static1.cpp

Q5] What is concept of Name Mangling

When we compile the code the compiler changes the ~~co~~ name of every function with 'mangled name' ie modified name.

When we overload the function, all the names of function are same but due to Name Mangling the compiler will change name of every function with new name as per pattern.

No concept of function overloading after code gets compiled.

Name Mangling

int Addition (int no1 int no2)

Addition@2ii = Addition @ 2 ; ;

initials of data types of every parameter

Refer: overloading1.cpp

86] How do we initialise the static characteristic of a class ?

static characteristics should be initialised outside the class using 'scope resolution operator' (::).

Declare the static data member using the 'static' keyword

Ex: static int var ;

int Myclass :: var = 0 ;

87] can we access private non-static characteristics of a class from static method ? Explain with example.

no we cannot access private nonstatic characteristics of a class directly, as static methods are not tied to any specific instance of the class.

we can access nonstatic characteristics indirectly using a reference pointer to an instance of class passed to the static method.

```
#include <iostream>
using namespace std;
class Myclass {
  private:
      int nonstaticvar;
      static int staticvar;
  public:
  Myclass (int value) : nonstaticvar (value) { }
  static void display () {
cout << " Static var " << staticvar << endl;
};
```

```
int MyClass :: Staticvar = 10;
int main() {
  My Class obj (20);
  MyClass :: display ();
  return 0;
}
```

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

Yes it is possible to create private static characteristics of class in c++.

These are defined with private Access specifier & are only accessible within the class.

They are not directly from outside the class. even using the class name or an object.

Static characteristics declared as private can only be accessed or modified through public member functions including static member functions.

```
#include <iostream>
using namespace std;
class Myclass {
Private:
static int privatestaticvar ;
Public:
Static void setprivatestaticvar (int value) {
   privatestaticvar = value;
}
```

```
static int getPrivatestaticvar() {
return privatestaticvar;
}
};
int MyClass :: privateStaticvar = 0;
int main() {
Myclass :: setprivatestaticvar(42);
cout << "Private static variable << Myclass :: getPrivateStatic-
var() << endl;
return 0;
}
```

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

They are created (allocated) only once when the program starts
They persist throughout the entire lifetime of the program, regardless of how many objects of the class are created or destroyed.
They are destroyed (deallocated) when the program ends.
Scope is determined by Access Specifier.

Q10] What is meant by static behaviour? Explain with example.
Program:- static3.cpp

The above thali class contains below things:-
1) Non-static characteristics (variable): Jamun & Rasgolla
2) Static Characteristics: Loncha
3) Non static behaviour: void display()
4) Static behaviour : static void show loncha()

In the above application 'showJoncha' is a static method.
To call this static method we can use the name of that class
Static method can be called without creating object.
Static method can access only static characteristics of a class
Static Behaviour refers to the use of the static keyword to define:-

▷ Static variable:
Retain value between function calls or share a single instance in a class

2> Static Member functions:
Belong to the class, not objects, & can access only static members