```dart
class Test {

 final int x;
 final int y;

 const Test(this.x,this.y){
   print("In const constructor");
 }
}

void main(){
  Test obj = Test(10,20);
 print(obj.x);
}
```

Output: program1.dart:6:2: Error: A const constructor can't have a body.
Try removing either the 'const' keyword or the body.

```
	const Test(this.x,this.y){
	^^^^^
```

Explanation:
 A constant constructor can’t have body, if it has body then it might be reassign the instance variable and it’s not for constant constructor behaviour.

Question 2:

```dart
class Employee {
  int? empId;
 String? empName;

 Employee(){}
 Employee(int empId, String empName){}
}

void main(){
 Employee obj = new Employee();
}
```

```
program2.dart:7:2: Error: 'Employee' is already declared in this scope.
        Employee(int empId, String empName){}
        ^^^^^^^^
program2.dart:6:2: Context: Previous declaration of 'Employee'.
        Employee(){}
        ^^^^^^^^
program2.dart:11:21: Error: Can't use 'Employee' because it is declared more than once.
        Employee obj = new Employee();
```

Explanation:
Everything in dart is an Object and it is not possible to exist same reference variable in the same scope.

```
class Demo {
  final int? x;
 final String? str;

 const Demo(this.x,this.str);
}

void main(){
 Demo obj1 = const Demo(10,"Core2web");
 print(obj1.hashCode);
  Demo obj2 = const Demo(10,"Biencaps");
 print(obj2.hashCode);
}
```

Output: 70127990
181121325

Explanation:
constant constructor is used to create compile time constant object and when it is called with same content then dart uses existing object to refer, this is for memory management of dart.

Question 4:

```
class Company {
 int empCount;
 String compName;

 Company(this.empCount, [this.compName = "Biencaps"]);

 void compInfo() {
   print(empCount);
   print(compName);
 }
}

void main() {
 Company obj1 = new Company(100);
 Company obj2 = new Company(200, "Pubmatic");

 obj1.compInfo();
 obj2.compInfo();
}
```

Output:
100
Biencaps
200
Pubmatic

Explanation: empCount is positional argument and compName is optional argument so empCount should be pass through constructor but compName is optional, if we pass it, it will take passed value otherwise default value, in this case “Biencaps”

Question 5:

```
class Company {
 int? x;
 String? str;

 Company(this.x, {this.str = "Core2web"});

 void compInfo() {
   print(x);
   print(str);
 }
}

void main() {
 Company obj1 = new Company(100);
 Company obj2 = new Company(200, "Incubator");

 obj1.compInfo();
 obj2.compInfo();
}
```

Output:

```
program5.dart:15:29: Error: Too many positional arguments: 1 allowed, but 2 found.
Try removing the extra positional arguments.
  Company obj2 = new Company(200, "Incubator");
                            ^
program5.dart:5:3: Context: Found this candidate, but the arguments don't match.
  Company(this.x, {this.str = "Core2web"});
```

Explanation:
Here, x is positional argument and str is named optional argument, and should be passed with name but here “Incubator” is passed as a positional argument so we get an error.

Question 6:

```
class Company {
 int? empCount;
 String compName;

 Company({this.empCount, this.compName = "Deloitte"});

 void compInfo() {
   print(empCount);
   print(compName);
  }
}

void main() {
 Company obj1 = new Company(empCount: 100, compName: "Veritas");
 Company obj2 = new Company(compName: "Pubmatic", empCount:
200);

 obj1.compInfo();
 obj2.compInfo();
}
```

Output:
100
Veritas
200
Pubmatic

Explanation:
empCount and compName are named argument and it is passed with name and can be passed with name random order.

Question 7:

```
class Point {
 int x;
 int y;
}

void main() {
 Point obj = Point();
}
```

Output:
```
program7.dart:2:7: Error: Field 'x' should be initialized because its type 'int' doesn't allow null.
  int x;
      ^
program7.dart:3:7: Error: Field 'y' should be initialized because its type 'int' doesn't allow null.
  int y;
```

Explanation:
To use any variable, it should be initialized first before use.

Question 8:

```dart
class Player {
 int? jerNo;
 String? pName;

 const Player(this.jerNo, this.pName);
}

void main() {
 Player obj = const (45, "Rohit");
}
```

```
program8.dart:9:22: Error: A value of type '(int, String)' can't be assigned to a variable of type 'Player'.
 - 'Player' is from 'program8.dart'.
  Player obj = const (45, "Rohit");
                     ^
program8.dart:5:9: Error: Constructor is marked 'const' so all fields must be final.
  const Player(this.jerNo, this.pName);
        ^
program8.dart:2:8: Context: Field isn't final, but constructor is 'const'.
  int? jerNo;
       ^
program8.dart:3:11: Context: Field isn't final, but constructor is 'const'.
  String? pName;
```

Explanation:
First Error: (int,String) can't be assign to the Player type reference

Second Error: If we declare the constant constructor, then all field must be final, to make the constructor constant means compile time constant.

Question 9:

```
int a = 10;

class Test {
 int x = 20;
 int y = 20;
 static num z = 30;

 Test(this.x, this.y, this.a);

 void fun() {
   print(a);
   print(x);
   print(y);
 }
}

void main() {
 Test obj = Test(10, 30, 40);
 obj.fun();
}
```

Output:
program9.dart:8:29: Error: 'a' isn't an instance field of this class.
Test(this.x, this.y, this.a);

Explanation:
Constructor is used for initialize the instance variable and not static variable. Here in constructor a is used and there is no instance variable with name a so it will give an error.

Question: 10

```
class Demo {
 Demo() {
   print("In demo");
 }

 factory Demo() {
   print("In demo factory");
   return Demo();
 }
}

void main() {
 Demo obj = new Demo();
}
```

Output:

```
program10.dart:6:11: Error: 'Demo' is already declared in this scope.
  factory Demo() {
          ^^^^
program10.dart:2:3: Context: Previous declaration of 'Demo'.
  Demo() {
  ^^^^
program10.dart:8:12: Error: Can't use 'Demo' because it is declared more than once.
    return Demo();
           ^
program10.dart:13:18: Error: Can't use 'Demo' because it is declared more than once.
  Demo obj = new Demo();
                 ^
```

Explanation:

Everything in dart is object and this is not possible that same reference pointing to different object in same frame. So it will give error for Demo.

Question 11:

```
class Test {
 Test._private() {
   print("In demo");
 }

 factory Test() {
   print("In demo factory");
   return Test._private();
 }
}

void main() {
 Test obj = new Test();
}
```

Output:
In demo factory
In demo

Explanation:

Here Test._private named private constructor is written and one factory constructor
So when we create an object factory will be called and inside that there is called for private constructor.