

# Constructor & Destructor

## Fill in the Blanks

1. constructor can be used to initialize the data members at the time of object creations. ✓
2. Name of the constructor of class main will be main ✓
3. constructor is the first member function to be executed when an object of that class is created. ✓
4. Copy constructor takes a reference to an object of the same class as an argument. ✓
5. Overloaded constructors differ in their argument ✓
6. Name of the Destructor is preceded by the tilt (~) symbol. ✓
7. The constructor of the global objects gets called before main ✓
8. constant object can be initialized only constructor

## State True or False

1. A constructor is automatically invoked when an object goes of scope false ✓
2. A constructor can be declared only in the public section. false ✓
3. A constructor can be called explicitly ~~true~~ false ✓
4. A function can be overloaded true ✓
5. Constructor and destructor must be defined inside the class. false
6. Copy constructor allocates memory for the data members dynamically. true
7. Constructor can also have default arguments true ✓

8. it is compulsory to place default arguments in the definition of a constructor. false ✓
9. It is possible to overload constructors. true ✓
10. destructor is called every time an object is created. false ✓
11. The order of invoking a destructor is the same as that of invoking a constructor. false ✓
12. The address of constructor and destructors can be accessed in programs. false ✓
13. An object with a constructor or destructor cannot be used as a member of a union. true ✓
14. constructor and destructor can be inherited. false ✓
15. Constructor can be virtual, but destructor can not be virtual. false ✓
16. Anonymous classes can also have constructors and destructors. false
17. Destructors can be overloaded. false ✓
18. Constructors and Destructors can return values false ✓

### Multiple Choice Questions

1. A constructor can be  
a) ~~virtual~~ b) Static c) Volatile d) none of these ✓
2. Which constructor does not initialize any data members  
a) Dummy b) Default c) Copy d) parameterised. ✓
3. Which constructor does not take any arguments?  
a) Dummy b) Default c) Copy d) parameterised. ✓
4. Which constructor creates a new object from an existing one?  
a) Dummy b) Default c) Copy d) parameterised. ✓
5. Which types of constructors is similar to a constructor that has all default arguments?  
a) Dummy b) Default c) Copy d) parameterised. ✓
6. If you write student s[30]; where student is the name of the class, how many times will the constructor function be invoked  
a) 29 b) 1 c) 30 d) 31 ✓

7. Ideally , in which section must constructor and destructor be declared.  
a) private   **b) public**   c) protected   d) any of these
8. How many destructors can have a class  
a) 0   **b) 1**   c) 2   d) N
9. Constant object can be initialized only by \_\_\_\_\_  
**a) Constructor**   b) member function   c) Main()   d) destructor
10. How many Anonymous objects can a class have at a particular time?  
a) 0   **b) 1**   c) 2   d) N
11. Constructor and destructor are automatically invoked by  
a) operating system   **b) compiler**   c) Main()   d) object.
12. Sample `s1 = s2;` will invoke \_\_\_\_\_ types of constructor?  
a) Dummy   b) Default   **c) Copy**   d) parameterised.

### Analyse the following Codes

```
1. #include<iostream>
using namespace std;
class A
{
    A() {
        cout<<"Constructor"<<endl;
    }
    ~A() {
        cout<<"Destructor"<<endl;
    }
};
main(){
    A obj1;
}
```

**o/p = Constructor   destructor**

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

```

class A
{
    int x;
    public:
    A(int a)    {
        x=a;
        cout<<"Constructor"<<endl;
    }
    ~A() {
        cout<<"Destructor"<<endl;
    }
};
main() {
A obj1;
}

```

**ERROR(BECAUSE OF CONSTRUCTOR TYPE)**

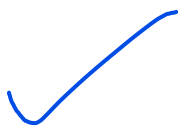


```

3.#include<iostream>
using namespace std;
class A
{
    int x;
    public:
    A(a) {
        x=a;
        cout<<"Constructor"<<endl;
    }
    ~A() {
        cout<<"Destructor"<<endl;
    }
};
main() {
A obj1;
}

```

**ERROR**



```

4.#include<iostream>
using namespace std;
class A

```

```

{
    int x;
    public:
    A(int a)    {
        x=a;
        cout<<"Constructor"<<endl;
    }
    void get_data() {
        cout<<"x="<<x<<endl;
    }
};
main() {
A obj1;
obj1.get_data();
}

```

ERROR(BECAUSE PARAMETRISED CONSTRUCTOR.should have default values)

```

5.#include<iostream>
using namespace std;
class abcd
{
    ~abcd() {
        cout << "welcome to vector india";
    }
public:
    abcd() {
        cout << "welcome to bangalore";
    }
};
int main()
{
    abcd vector;
}

```

welcome to bangalore

ERROR

```

6.#include<iostream>
using namespace std;
class Point

```

```

{
    Point() {
        cout << "Constructor called";
    }
};
int main() {
    Point t1;
}

```

**NOTHING DISPLAYED** . ERROR ✓

```

7. #include<iostream>
using namespace std;
class Ex
{
public:
    void ~Ex() {
        cout<<"Destroying the object";
    }
};
int main() {
    Ex abcd;
}

```

**ERROR (BECAUSE OF RETURN TYPE OF DESTRUCTOR)** ✓

```

8. #include<iostream>
using namespace std;
class Exam
{
public:
    Exam() {
        cout << "Constructor called ";
    }
}

```

```
};
int main()
{
    Exam Ex1, Ex2;
}
```

CONSTRUCTOR CALLED CONSTRUCTOR CALLED



```
9.#include<iostream>
using namespace std;
class ample
{
public:
    int a;
    int b;
};
```

```
int main()
{
    ample Ex1 = {10, 20};
    cout << "a = " << Ex1.a << ", b = " << Ex1.b;
}
```

a=10, b=20

