2 CLASSES

This chapter takes you across the nooks and corners of classes. This concept is the primary necessity for Object Oriented Programming. After going through this chapter you will be able to handle classes, objects and any related concepts easily.

1. What is the output of the code? (L1)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
    int marks;
void percent()
{ cout<< marks*10; }
    }S;
    int main()
    { S.marks=9;
    S.percent(); }
```

(a) 90 (b) 9

(c) Error (d) Unpredictable

Answer: (a) `'S'` is the object / instance of the class. It can access the `'public'` members of the class. `'public'` is the access specifier. Classes have two more specifiers `'private'` and `'protected'`. The variables are called `'data members'` and the functions are called `'member functions'`.

2. What is the output of the code? (L1)

```
    #include<iostream>
    using namespace std;
    class student
    { private:
    int marks;
void percent()
{ cout<< marks*10; }
    }S;
    int main()
    { S.marks=9;
    S.percent(); }
```

(a) 90 (b) 9

(c) Error (d) Unpredictable

Answer: (c) `'private'` access specifier means the data that is to be hidden from other unrelated info and user. Trying to access `'private'` data and methods throws errors. It is accessible only by the members of the class and not even the objects.

3. What will be the error thrown by the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { private:
    int marks;
void percent()
{ cout<< marks*10; }
    }S;
    int main()
    { S.marks=9; }
```

(a) `'int student::marks'` is private

(b) within this context `S.marks=9;`

(c) `'student::marks'` not defined

(d) Both 'a' and 'b'

(e) Both 'd' and 'b'

(a) 4 (b) 0
(c) Unpredictable (d) None of the above

Answer: (a) Even though the members are not accessible by the object, each object will have a specific value of the variable. So it will be allocated.

9. What is the output of the code? (L2)

```
    #include<iostream>
    using namespace std;
    class student
    { protected:
int marks;
void percent()
{ cout<< marks*10; }
    }S;
    int main()
    { cout<<sizeof(S); }
```

(a) 4 (b) 0
(c) Unpredictable (d) None of the above

Answer: (a) The same here.

10. What is the output of the code? (L1)

```
    #include<iostream>
    using namespace std;
    class student
    { char section;
public:
int marks;
void percent()
{ cout<< marks*10; }
    }S;
    int main()
    { cout<<sizeof(S); }
```

(a) 4 (b) 5
(c) Unpredictable (d) None of the above

Answer: (b) This works normally.

11. What is the output of the code? (L1)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int marks;
char section;
    }S;
    int main()
    { S.marks=8;
S.section = 'a';
cout<<S.marks<< " "<<S.section;
}
```

(a) 8 a (b) Unpredictable
(c) Error (d) None of the above

Answer: (a) Unnecessary blank lines within the 'class' doesn't change the specifiers to default ('private').

12. What is the output of the code? (L2)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int marks;
private:
char section;
    }S;
    int main()
    { S.marks=8;
S.section = 'a';
cout<<S.marks<< " "<<S.section; }
```

(a) 8 a (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) This works exactly as expected.

13. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int marks;
private:
int m;
public:
char section;
```

```
        }S;
        int main()
        { S.marks=8;
    S.section = 'a';
    cout<<S.marks<< " "<<S.section; }
```

(a) 8 a (b) Unpredictable
(c) Error (d) None of the above

Answer: (a) Same access specifier can be given multiple times.

14. What is the output of the code? (L3)

```
        #include<iostream>
        using namespace std;
        class student
        { private:
    int marks;
    public:
    int m;
    private:
    char section;
        }S;
        int main()
        { S.m=8;
        cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (a) This also works the same.

15. What is the output of the code? (L3)

```
        #include<iostream>
        using namespace std;
        class student
        { private:
    public:
    int m;
        }S;
        int main()
        { S.m=8;
    cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (a) A specifier block can be empty.

16. What is the output of the code? (L4)

```
        #include<iostream>
        using namespace std;
        class student
        { private:
    public:
    int m;
        }S;
        int main()
        { S.m=8;
    cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (a) Having specifiers in continuous lines doesn't make them interdependent. By default it declares the first block as empty.

17. What is the output of the code? (L4)

```
        #include<iostream>
        using namespace std;
        class student
        { private, public:
    int m;
        }S;
        int main()
        { S.m=8;
    cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) This is not legal. But it is to be noted that still the specifiers exist as keywords.

18. What will be the error thrown by the code? (L4)

```
        #include<iostream>
        using namespace std;
        class student
        { private, public:
    int m;
        }S;
        int main()
        { S.m=8;
    cout<<S.m; }
```

(a) Expected `':'` before `','` token in `'private, public:'`
(b) Expected unqualified-id before `','` token
(c) `'class student'` has no member named 'm'
(d) All of the above

Answer: (d) It is to be noted that `'c'` will be thrown twice because it is accessed twice.

19. What is the output of the code? (L4)

```
        #include<iostream>
        using namespace std;
        class student
        { private / public:
    int m;
        }S;
        int main()
        { S.m=8;
    cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) The same happens here.

20. What is the output of the code? (L4)

```
        #include<iostream>
        using namespace std;
        class student
        { [private,public]:
    int m;
        }S;
        int main()
        { S.m=8;
    cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) There is no way to specify multiple specifiers together.

21. What is the output of the code? (L2)

```
        #include<iostream>
        using namespace std;
        class student
        { private:public:
    int m;
        }S;
        int main()
        { S.m=8;
    cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (a) This works normally as specified above.

22. What is the output of the code? (L2)

```
        #include<iostream>
        using namespace std;
        class student
        { public:
    int m;
    privte.int s;
        }S;
        int main()
        { S.s=8;
    cout<<S.s; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) This is not allowed. It gives the same error.

23. What is the output of the code? (L3)

```
        #include<iostream>
        using namespace std;
        class student
        { public:
    m;
        }S;
        int main()
        { S.m=8;
    cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) This is not possible.

24. What will be the error thrown by the code? (L4)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
m;
    }S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) `'m'` does not name a type
(b) Invalid declaration of class member `'m'`
(c) `'class student'` has no member named `'m'`
(d) Both 'a' and 'c'
(e) Both 'b' and 'c'

Answer: (d) It doesn't recognise the variable as member itself. It treats `'m'` as a type.

25. What is the output of the code? (L4)

```
    #include<iostream>
    using namespace std;
    int m;
    class student
    { public:
m;
    }S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) The same repeats.

26. What is the output of the code? (L4)

```
    #include<iostream>
    using namespace std;
    int m;
    class student
    { public:
::m;
    }S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) Global variables cannot be brought into class like this using scope resolution operator.

27. What will be the error thrown by the code? (L4)

```
    #include<iostream>
    using namespace std;
    int m;
    class student
    { public:
::m;
    }S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) Using-declaration for non-member at class scope
(b) Invalid usage of `'::'` operator
(c) `'class student'` has no member named `'m'`
(d) Both 'b' and 'c'
(e) Both 'a' and 'c'

Answer: (e) Here `'m'` is not a member of a class. So it cannot be declared again inside the class using `'::'` operator.

28. What will be the error thrown the code? (L4)

```
    #include<iostream>
    using namespace std;
    int m;
    class student
    { public:
    int m;
}S;
int main()
{ S.m=8;
m=0;
cout<<S.m<< " "<<m; }
```

(a) Using-declaration for non-member at class scope
(b) Redeclaration of variable 'm'
(c) Both 'a' and 'b'
(d) None of the above

Answer: (d) It throws no error. It is allowed in classes. The output will be '8 0'.

29. What is the output of the code? (L4)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int m;
    }m;
    int main()
    { m.m=8;
cout<<m.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (a) Class object can be same as a member variable.

30. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { int m;
public:
int m;
    }S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) 8 (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) This is not allowed as the members will collide.

31. What will be a part of the error thrown by the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { int m;
public:
int m;
    }S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) Redeclaration of 'int student::m'
(b) 'int student::m' is private
(c) Both 'a' and 'b'
(d) None of the above

Answer: (c) Before declaring 'm' as public it is declared as private.

32. What will be the error thrown by the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int m;
int m;
    }S;
    int main()
{ S.m=8;
cout<<S.m; }
```

(a) Redeclaration of 'int student::m'
(b) 'int student::m' is declared as public only
(c) Both 'a' and 'b'
(d) None of the above

(a) 8 (b) Error
(c) 0 (d) None of the above

Answer: (b) The statement `'S.m;'` has issues.

38. What will be the error thrown by the code? (L4)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int m;
}S;
    S.m;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) Invalid type `'S'`
(b) Useless declaration
(c) `'S'` does not name a type
(d) None of the above

Answer: (c) Compiler treats it as a new variable declaration, and throws errors.

39. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int m;
}S;
    int m;
    int main()
    { m=0;
S.m=8;
cout<<S.m<< " "<<m; }
```

(a) 8 0 (b) Error
(c) 0 0 (d) None of the above

Answer: (a) Declaring variables after the class with member names doesn't affect the declaration. Only variables that are defined outside and later declared inside class gives errors.

40. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int m=9;
}S;
    int main()
    { cout<<S.m; }
```

(a) 9 (b) Error
(c) 0 (d) None of the above

Answer: (a) It gives a warning "non-static data member initializers only available with `-std=c++11` or `-std=gnu++11` [enabled by default]". Such initialisations can be done only to static members (that to outside the class usually). But this is available in the newer version compilers. So it is enabled by default. The explanation of static variables will be done in upcoming chapters.

41. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int m=9;
}S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) 8 (b) 9
(c) Error (d) 0

Answer: (a) It works normally.

42. What is the output of the code? (L1)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int m;
}S,S1;
```

```
    int main()
    { S.m=8;
cout<<S.m<< " "<<S1.m; }
```

(a) 8 8 (b) 8 0
(c) Error (d) Unpredictable

Answer: (b) This is a way of creating multiple objects.

43. What is the output of the code? (L1)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
int m=9;
}S,S1;
    int main()
    { S.m=8;
cout<<S.m<< " "<<S1.m; }
```

(a) 8 8 (b) 8 9
(c) Error (d) Unpredictable

Answer: (b) The object that is not initialised gets default value.

44. What is the output of the code? (L1)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
    int m; };
    int main()
    { student S;
S.m=8;
cout<<S.m; }
```

(a) 8 (b) 0
(c) Error (d) Unpredictable

Answer: (a) This is a way of creating objects inside main.

45. What is the output of the code? (L1)

```
    #include<iostream>
    using namespace std;
    class student;
    class student
    { public:
    int m; };
    int main()
    { student S;
S.m=8;
cout<<S.m; }
```

(a) 8 (b) 0
(c) Error (d) Unpredictable

Answer: (a) The new statement `'class student;'` is called forward declaration of the class. This is used to inform the compiler that such a class will be defined later. This is useful when declaring multiple classes and communicating amongst them. But it is mandatory that all the classes should be defined before `'main()'`. Else it will throw errors of incomplete type.

46. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
    int m;
}S;
    int main()
    { student S;
S.m=8;
cout<<S.m; }
```

(a) 8 (b) 0
(c) Error (d) Unpredictable

Answer: (a) Actually the object `'S'` inside main is local variable. And the object after the class declaration is global.

47. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
    int m;
}S;
    student S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) 8 (b) 0
(c) Error (d) Unpredictable

Answer: (c) Now the scope clashes and throws error.

48. What will be the error thrown by the code? (L4)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
    int m;
}S;
    student S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) Redefinition of `'student S'`
(b) Redeclaration of `'student S'`
(c) Redefinition of instance `'student S'`
(d) Redeclaration of instance `'student S'`

Answer: (a) It is called object definition.

49. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
    int m;
student S;
}S;
    int main()
{ S.m=8;
cout<<S.m; }
```

(a) 8 (b) 0
(c) Error (d) Unpredictable

Answer: (c) The object cannot be inside the class itself, as it is not completely defined.

50. What will be the error thrown by the code? (L4)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
    int m;
student S;
}S;
    int main()
    { S.m=8;
cout<<S.m; }
```

(a) Undefined type `'student'`
(b) Field `'S'` has incomplete type
(c) Both 'a' and 'b'
(d) None of the above

Answer: (b) The compiler knows `'student'` is a class and is not complete till now.

51. What is the output of the code? (L3)

```
    #include<iostream>
    using namespace std;
    class student
    { public:
    int m; }
S;
int main()
{ S.m=8;
cout<<S.m; }
```

(a) 8 (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) It works normally. Even though `'S;'` is given in next line, it will be considered as continuity.

57. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m;
      class subject
      { public:
    char grade;
      }sb;
}S;
int main()
{ sb.grade= 'A';
      cout<<sb.grade; }
```

(a) `'sb'` not declared
(b) `'sb'` was not declared in this scope
(c) Unknown variable `'sb'`
(d) `'sb'` has no member `'grade'`

Answer: (b) This is because `'sb'` is considered a member of `'class student'`.

58. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m;
      class subject
      { public:
    char grade;
      }sb;
}S;
int main()
{ S.sb.grade= 'A';
      cout<<S.sb.grade; }
```

(a) A (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) This works right as `'sb'` is treated as an object of `'class student'`.

59. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m;
      class subject
      { public:
    char grade; };
}S;
int main()
{ subject sb;
      sb.grade= 'A';
      cout<<sb.grade; }
```

(a) A (b) Error
(c) Unpredictable (d) None of the above

Answer: (b) The class defined inside another class is not accessible by other classes.

60. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m;
      class subject
      { public:
    char grade; };
}S;
int main()
{ subject sb;
      sb.grade= 'A';
      cout<<sb.grade; }
```

(a) `'subject'` was not declared in this scope
(b) Expected `';'` before `'sb'`
(c) `'sb'` was not declared in this scope
(d) All of the above

Answer: (d) `'subject'` is not accessible by outside.

61. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m;
      class subject
      { public:
    char grade; };
}S;
int main()
{ S.subject sb;
      sb.grade= 'A';
      cout<<sb.grade; }
```

(a) A (b) Unpredictable
(c) Error (d) None of the above

Answer: (c) This is not a legal way of creating objects for class that is inside other classes.

62. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m;
      class subject
      { public:
    char grade; };
}S;
int main()
{ S.subject sb;
      sb.grade= 'A';
      cout<<sb.grade; }
```

(a) `'sb'` was not declared in this scope
(b) Expected `';'` before `'sb'`
(c) Invalid use of `'class student::subject'`
(d) Both 'a' and 'b'
(e) 'a', 'b' and 'c'

Answer: (e) Objects for classes defined inside other classes cannot be created outside the outer class.

63. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m;
      class student
      { public:
    char grade; }sb;
}S;
int main()
{ S.m=9;
      cout<<S.m; }
```

(a) Redeclaration of `'class student'`
(b) `'class student'` incomplete
(c) Invalid use of constructor
(d) `'student::student'` has the same name as the class in which it is declared
(e) No error

Answer: (d) class inside a class with same name is not possible.

64. What will be a part of the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m; }S;
class student
{ public:
    char grade; }sb;
int main()
{ S.m=9;
      cout<<S.m; }
```

(a) Redeclaration of `'class student'`
(b) Redefinition of `'class student'`

(c) Both 'a' and 'b'
(d) No error will be thrown

Answer: (b) Here `'class student'` is defined twice. This throws further errors in declaration of object of the duplicate classes and its accesses.

65. What is the output of the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m; }S;
class student
{ public:
   int m; }sb;
int main()
{ S.m=9;
     cout<<S.m; }
```

(a) 9 (b) 0
(c) Error (d) Unpredictable

Answer: (c) Even though both the definitions are completely identical, it is considered a new definition and the same errors will be thrown.

66. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m; }S;
class subject
{ public:
   int g; }S;
int main()
{ S.m=9;
     cout<<S.m; }
```

(a) 9 (b) 0
(c) Error (d) Unpredictable

Answer: (c) Multiple classes cannot have same object name.

67. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m; }S;
class subject
{ public:
   int g; }S;
int main()
{ S.m=9;
     cout<<S.m; }
```

(a) Redeclaration of `'student S'` as `'subject S'`
(b) Redefinition of `'student S'` as `'subject S'`
(c) Conflicting declaration `'subject S'`,`'S'` has a previous declaration as `'student S'`
(d) None of the above

Answer: (c) The compiler points the conflict.

68. What is the output of the code? (L2)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m; }S;
int main()
{ class subject
     { public:
int g; }sb;
     sb.g=9;
     cout<<sb.g; }
```

(a) 9 (b) 0
(c) Error (d) Unpredictable

Answer: (a) This is a local class declaration. It is accessible only inside main.

69. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m; }S;
int main()
{ class student
      { public:
      int g; }sb;
      sb.g=9;
      cout<<sb.g; }
```

(a) 9 (b) 0
(c) Error (d) Unpredictable

Answer: (a) The local `'class'` doesn't clash with the global class.

70. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m; }S;
int main()
{ class student
      { public:
         int m; }S;
      S.m=9;
      cout<<S.m; }
```

(a) 9 (b) 0
(c) Error (d) Unpredictable

Answer: (a) Even now the scope plays the separating role.

71. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m; }S;
int main()
{ class student
      { public:
         int m; }S;
      S.m=9;
      cout<<S.m<< " "<<::S.m; }
```

(a) 9 0 (b) 0 9
(c) Error (d) None of the above

Answer: (a) `'::'` is used to access the global scope object.

72. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m; }S;
int main()
{ class student
      { public:
   int m; }Sb;
      Sb.m=9;
      cout<<Sb.m<< " "<<::S.m; }
```

(a) 9 0 (b) 0 9
(c) Error (d) None of the above

Answer: (a) Even if the name is not clashing, `'::'` can be used to point to global scope variable.

73. What is the output of the code? (L1)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<<m;} }S;
int main()
{ S.m=9;
      S.fun();}
```

(a) 9 (b) 0
(c) Error (d) None of the above

(a) 9 (b) 0
(c) Error (d) None of the above

Answer: (c) This is not the legal way of defining functions outside.

79. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
    int m;
    void fun(); }S;
student::void fun()
{ cout<<m;}
int main()
{ S.m=9;
      S.fun();}
```

(a) Expected function name before `'void'`
(b) Qualified-id `'void'` for `'function'`
(c) Expected unqualified-id before `'void'`
(d) None of the above

Answer: (c) `'void'` is treated as function name.

80. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
void student::fun()
{ cout<<m;}
class student
{ public:
    int m;
    void fun(); }S;
int main()
{ S.m=9;
      S.fun();}
```

(a) 9 (b) 0
(c) Error (d) None of the above

Answer: (c) As student is not defined before the function definition, it gives errors.

81. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
void student::fun()
{ cout<<m;}
class student
{ public:
    int m;
    void fun(); }S;
int main()
{ S.m=9;
      S.fun();}
```

(a) `'student'` has not been declared
(b) Unknown type `'student'`
(c) `'m'` was not declared in this scope
(d) Both 'a' and 'c'
(e) Both 'b' and 'c'

Answer: (d) `'student'` cannot be accessed before it is defined.

82. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
void fun()
{ cout<<m;}
class student
{ public:
    int m;
    void fun(); }S;
int main()
{ S.m=9;
      S.fun();}
```

(a) 9 (b) 0
(c) Error (d) None of the above

Answer: (c) Global functions and member functions with same name doesn't map together. The class member `'fun'` is not mapped to definition of global `'fun'`.

83. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
void fun()
{ cout<<m;}
class student
{ public:
   int m;
   void fun(); }S;
int main()
{ S.m=9;
     S.fun();}
```

(a) Invalid overwriting of `'student::fun'`
(b) Undefined reference to `'student::fun()'`
(c) Invalid overriding of 'student::fun'
(d) None of the above

Answer: (b) The member function `'fun'` is not defined.

84. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<< "hai";}
}S;
void student::fun()
{ cout<<m;}
int main()
{ S.m=9;
     S.fun();}
```

(a) 9 (b) `hai`
(c) `hai9` (d) Error

Answer: (d) One member should be defined only once.

85. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<< "hai";}
}S;
void student::fun()
{ cout<<m;}
int main()
{ S.m=9;
     S.fun();}
```

(a) `'fun'` redefined
(b) Redefinition of `'void fun'`
(c) Redefinition of `'void student::fun()'`
(d) None of the above

Answer: (c) member will be identified with the class name only.

86. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<< "hai";}
}S;
void student::fun();
int main()
{ S.m=9;
     S.fun();}
```

(a) Redeclaration of `'void student::fun()'`
(b) Redeclaration of class member `'void fun'`

```
{ public:
   int m; }S;
void student::fun();
void student::fun()
{ cout<<m;}
int main()
{ S.m=9;
     S.fun();}
```

(a) 9 (b) 0
(c) Unpredictable (d) Error

Answer: (d) Even this doesn't work. There is no way to add a member after declaration of class.

92. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<<m;}
}S;
int student::m;
int main()
{ S.m=9;
     S.fun();}
```

(a) 9 (b) 0
(c) Unpredictable (d) Error

Answer: (d) It says "`'int student::m'` is not a static member of `'class student'`". Only static data members can be given like this. `'static'` members will be explained in detail in a later chapter.

93. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<<m;}
}S;
int main()
{ student s;
     S.m=9;
     s.fun();}
```

(a) 9 (b) 0
(c) Unpredictable (d) Error

Answer: (c) Data member's values are closed to the object. Undefined data members give some garbage values. Unlike `'C'` here the undefined values are not initiated to 0.

94. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<<m;}
}S;
int main()
{ student s=S;
     S.m=9;
     s.fun();}
```

(a) 9 (b) 0
(c) Unpredictable (d) Error

Answer: (b) `'S'` gets the value 9 only after equating to `'s'`. So `'s'` gets 0. Object can be equated to other objects.

95. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<<m;}
}S;
int main()
```

```
{ student s;
     S.m=9;
     s=S;
     s.fun();}
```

(a) 9 (b) 0
(c) Unpredictable (d) Error

Answer: (a) This works as expected.

96. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<<m;}
}S;
int main()
{ S.m=9;
     student s(S);
     s.fun();}
```

(a) 9 (b) 0
(c) Unpredictable (d) Error

Answer: (a) This is another way of creating object from another object. Its working is explained in a later chapter.

97. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class student
{ public:
   int m;
   void fun()
   { cout<<m;}
}S;
int main()
{ S.m=9;
     student s(S);
     s.m=7;
     S.fun();}
```

(a) 9 (b) 7
(c) Unpredictable (d) Error

Answer: (a) Change in new objet doesn't reflect on the referenced object value.

98. What is the output of the code? (L2)

```
#include<iostream>
using namespace std;
class student
{ protected:
    int m;
     public:
    void fun()
    { m=9;
    cout<<m;}
}S;
int main()
{ S.fun();}
```

(a) 9 (b) 0
(c) Unpredictable (d) Error

Answer: (a) A class member can access 'protected' members.

99. What is the output of the code? (L2)

```
#include<iostream>
using namespace std;
class student
{ private:
    int m;
     public:
    void fun()
    { m=9;
    cout<<m;}
}S;
int main()
{ S.fun();}
```

(a) 9 (b) 0
(c) Unpredictable (d) Error

Answer: (a) A class member can access 'private' members.

```
{ public:
   class sub
   { public:
     void fun()
     { cout<< "sub-class";}
   }D;
}S;
int main()
{ S.D.fun();}
```

(a) `sub-class` (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) Now the function can be accessed.

105. What is the output of the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ class sub
   { public:
        void fun()
        { cout<< "sub-class";}
   };
   public: sub D;
}S;
int main()
{ S.D.fun();}
```

(a) `sub-class` (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) This is also correct. Even though the class is defined in private scope, the object of the class is in public scope. So it can access it easily.

106. What is the output of the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ class sub
   { public:
        void fun();
   };
public:
     void sub::fun()
    { cout<< "sub-class";}
     sub D;
}S;
int main()
{ S.D.fun();}
```

(a) `sub-class` (b) Error
(c) Unpredictable (d) None of the above

Answer: (b) "cannot define member function `'student::sub::fun'` within student using `'::'` operator.

107. What is the output of the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ class sub
   { public:
        void fun();
   };
public:
     sub D;
}S;
void student::sub::fun()
     { cout<< "sub-class";}
int main()
{ S.D.fun();}
```

(a) `sub-class` (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) This is a proper definition. As the class is confined to scope of inner class, while explicit definition of the function, it requires the whole hierarchy.

108. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ class sub
   { public:
    void fun()
          { cout<< "sub-class";}
    };
public:
     sub D;
}S;
int main()
{ S.fun();}
```

(a) Unknown member `'fun()'`
(b) Class student has no member named `'fun'`
(c) `'fun'` doesn't belong to the scope of class student
(d) `'student::sub::fun'` cannot be accessed from class student

Answer: (b) The compiler doesn't care about the existence of the function. It just checks if the function called is available in the current scope.

109. What is the output of the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ class sub
    { public:
          void fun();
    };
public:
      sub D;
      viod game
    { D.fun(); }
}S;
void student::sub::fun()
      { cout<< "sub-class";}
int main()
{ S.game();}
```

(a) `sub-class` (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) A function of the outer class can access the variable declared in the class. Here the object of the inner class is treated as any member of the class.

110. What is the output of the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
      class sub
    { public:
    void fun()
          { cout<< "sub-class"; }
    };
    void game()
    { sub d;
    d.fun(); }
      }S;
int main()
{ S.game(); }
```

(a) `sub-class` (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) Here, the object `'d'` created inside `'student::fun'` is temporary. It will be deleted once the function has completed execution.

111. What is the output of the code? (L4)

```
#include<iostream>
using namespace std;
class student
{ public:
      class sub
    { public:
    void fun()
          { cout<<
"sub-class"; }
    };
    void fun()
    { sub d;
{ public:
     int *f;
     int fun()
     { return 3;}
}S;
int main()
{ S.f = &(S.fun());
     cout<<*(S.f); }
```

(a) Invalid access for address '&' of a constant
(b) Lvalue required as unary '&' operand
(c) Invalid operand for unary operator '&'
(d) None of the above

Answer: (b) Compiler expects only lvalues and expressions.

117. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class stud
{ public:
     int f;
     void fun()
     { cout<<f;}
}S;
int main()
{ Stud *C;
     C = &S;
     *C.f = 3;
     *C.fun(); }
```

(a) 3 (b) Error
(c) Unpredictable (d) None of the above

Answer: (b) Objects can have pointers, but it should not be used like this for accessing.

118. What will be the error thrown by the code? (L4)

```
#include<iostream>
using namespace std;
class stud
{ public:
     int f;
     void fun()
     { cout<<f;}
}S;
int main()
{ Stud *C;
     C = &S;
     *C.f = 3;
     *C.fun(); }
```

(a) Expected '(' before 'C' in '*C.f = 3'
(b) Invalid lvalue for '*' operand
(c) Invalid use of '.' For pointer of class 'stud'
(d) None of the above

Answer: (d) It throws "request for member 'f' in 'C', which is of pointer type 'stud*' (maybe you meant to use '->' ?)". It is the syntax for pointer of object.

119. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class stud
{ public:
     int f;
     void fun()
     { cout<<f;}
}S;
int main()
{ Stud *C;
     C = &S;
     C->f = 3;
     C->fun(); }
```

(a) 3 (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) The '->' is called arrow operator. This must be used for pointers of class objects.

120. What is the output of the code? (L3)

```
#include<iostream>
using namespace std;
class stud
{ public:
```

```
int main()
{ S.f = 2;
     ((stud *)this)->fun(); }
```

(a) 2 (b) Error
(c) Unpredictable (d) None of the above

Answer: (b) 'this' pointer can be used only inside a class.

125. What will be the error thrown by the code? (L5)

```
#include<iostream>
using namespace std;
class stud
{ public:
     int f;
     void fun()
     { cout<<f;}
}S;
int main()
{ S.f = 2;
     ((stud *)this)->fun(); }
```

(a) Invalid use of 'this' in non-member function
(b) 'this' used in a non-member
(c) 'this' usage doesn't map to a class
(d) None of the above

Answer: (a) 'this' usage must be inside a member function only.

126. What is the output of the code? (L5)

```
#include<iostream>
using namespace std;
class stud
{ public:
     int f;
}S;
class blo
{ public:
     void fun()
     { cout<<((stud *)this)->f;}
}D;
int main()
{ cout<< "this in different class"; }
```

(a) This is different class
(b) Error
(c) Unpredictable
(d) None of the above

Answer: (a) 'this' pointer can also be used in a different and unrelated class.

127. What is the output of the code? (L5)

```
#include<iostream>
using namespace std;
class stud
{ public:
     int f;
}S;
class blo
{ public:
   void fun()
     { cout<<((stud *)this)->f;}
}D;
int main()
{ S.f = 2;
     D.fun(); }
```

(a) 2 (b) Error
(c) Unpredictable (d) None of the above

Answer: (d) It prints '0'. This is because, '2' is given to specific object 'S'. But 'this' pointer calls the general function and not the specific to 'S'.

128. What is the output of the code? (L5)

```
#include<iostream>
using namespace std;
class stud
{ public:
     static int f;
}S;
int stud::f = 3;
class blo
{ public:
        void fun()
        { cout<<((stud *)this)->f;}
}D;
int main()
{ D.fun(); }
```

(a) 3 (b) Error
(c) Unpredictable (d) None of the above

Answer: (a) 'static' variable is common for the class so gives the value. The details and usage of 'static' variable is discussed in upcoming chapters.

129. What is the output of the code? (L5)

```
#include<iostream>
using namespace std;
class stud
{ public:
        int f;
        void fun()
        { cout<<f; }
}S;
class blo
{ public:
        void fun()
        { ((stud *)this)->fun();}
}D;
int main()
{ D.fun(); }
```

(a) 3 (b) Error
(c) Unpredictable (d) None of the above

Answer: (c) When an object is specifically declared and generally called, the variable gets garbage value.

130. What is the output of the code? (L5)

```
#include<iostream>
using namespace std;
class stud
{ public:
        int f;
        void fun()
        { cout<<f; }
}S;
class blo
{ public:
        void fun(int a)
        { if(a%2 == 0)
            ((stud *)
this)->fun();
     else
            cout<<((stud *)this)->fun(); }
}D;
int main()
{ D.fun(7); }
```

(a) 3 (b) Error
(c) Unpredictable (d) None of the above

Answer: (b) It tries to get a value from a 'void' function and print it.

131. What is the output of the code? (L5)

```
#include<iostream>
using namespace std;
class stud
{ public:
        int f;
        void fun()
        { cout<<f; }
}S;
class blo
{ public:
        void fun(int a)
        { if(a%2 == 0)
            ((stud *)
this)->fun();
     else
            cout<<((stud *)this)->fun(); }
}D;
int main()
{ D.fun(6); }
```

135. What is the output of the code? (L5)

```
#include<iostream>
using namespace std;
class blo
{ public:
     int b;
     void fun(int a)
     { cout<<((stud *)this)->s->b; }
}S;
class stud
{ public:
     blo *s;
}D;
int main()
{ S.b = 2;
     S.fun(); }
```

(a) 2 (b) Error
(c) Unpredictable (d) None of the above

Answer: (b) Here usage of `'stud'` causes the error as it is not yet defined.

136. What will be the error thrown by the code? (L5)

```
#include<iostream>
using namespace std;
class blo
{ public:
     int b;
     void fun(int a)
     { cout<<((stud *)this)->s->b; }
}B;
class stud
{ public:
     blo *s;
}D;
int main()
{ S.b = 2;
     S.fun(); }
```

(a) Expected `')'` before `'this'` in `'((stud *)this'`
(b) Expected primary-expression before `')'` token in `'((stud *)'`
(c) `'stud'` was not declared in this scope
(d) None of the above

Answer: (d) It throws all the three. This is because the compiler treats `'stud'` as an undefined variable and not a function.

137. What is the output of the code? (L5)

```
#include<iostream>
using namespace std;
class stud;
class blo
{ public:
     int b;
     void fun(int a)
     { cout<<((stud *)this)->s->b; }
}S;
class stud
{ public:
     blo *s;
}D;
int main()
{ S.b = 2;
     S.fun(); }
```

(a) 2 (b) Error
(c) Unpredictable (d) None of the above

Answer: (b) Even though `'stud'` has been declared earlier it is not yet defined. `'blo'` cannot access a member of 'stud' even before it is defined.

138. What will be the error thrown by the code? (L5)

```
#include<iostream>
using namespace std;
class stud;
class blo
{ public:
```