**Classes**

**Writer:** Sayak Haldar

**1. What does your class can hold?**

a) data

b) functions

c) both data & functions

d) none of the mentioned

**Answer: c**

Explanation: The classes in c++ are used to manipulate both data and functions.

**2. How many specifiers are present in access specifiers in class?**

a) 1

b) 2

c) 3

d) 4

**Answer: c**

Explanation: There are three types of access specifiers. They are public, protected and private.

Unlike java, where there is another access specifier. Which is package default

**3. Which is used to define the member of a class externally?**

a) :

b) ::

c) #

d) none of the mentioned

**Answer: b**

**4. Which other keywords are also used to declare the class other than class?**

a) struct

b) union

c) object

d) both struct & union

**Answer: d**

Explanation: Struct and union take the same definition of class but differs in the access techniques.

Now, this is little twisting. Right? Because, if you think, you will immediately spot out one thing: That is all of them are derived datatype.

Now, unlike c, in c++, structures and unions have function declaration with in it. You can define those functions with in the structure/union. You can define them outside the structure/union using scope operator.

But, there is one necesssary difference. The members of structures and unions are by default public, whereas, in case of class, those are by default private.

**5. What is the output of this program?**

#include <iostream>

using namespace std;

class rect

{

int x, y;

public:

void val (int, int);

int area ()

{

return (x \* y);

}

};

void rect::val (int a, int b)

{

x = a;

y = b;

}

int main ()

{

rect rect;

rect.val (3, 4);

cout << "rect area: " << rect.area();

return 0;

}

a) rect area:12

b) rect area: 12

c) rect area:24

d) none of the mentioned

**Answer: b**

Explanation: In this program, we are calculating the area of rectangle based on given values.

Output:

$ g++ class.cpp

$ a.out

rect area: 12

**6. What is the output of this program?**

#include <iostream>

using namespace std;

class CDummy

{

public:

int isitme (CDummy& param);

};

int CDummy::isitme (CDummy& param)

{

if (&param == this)

return true;

else

return false;

}

int main ()

{

CDummy a;

CDummy \*b = &a;

if (b->isitme(a))

{

cout << "execute";

}

else

{

cout<<"not execute";

}

return 0;

}

a) execute

b) not execute

c) none of the mentioned

d) both execute & not execute

**Answer: a**

Explanation: In this program, we are just pointing the pointer to a object and printing execute if it is correctly pointed.

Output:

$ g++ class1.cpp

$ a.out

**7. Which of the following is a valid class declaration?**

a) class A { int x; };

b) class B { }

c) public class A { }

d) object A { int x; };

**Answer: a**

**8. The fields in the class in c++ program are by default**

a) protected

b) private

c) public

d) none of the mentioned

**Answer: b**

**9. Constructors are used to**

a) initalize the objects

b) construct the data members

c) both initalize the objects & construct the data members

d) none of the mentioned

**Answer: a**

Once the object is declared means, the constructor are also declared by default.

**10. When struct is used instead of the keyword class means, what will happen in the program?**

a) access is public by default

b) access is private by default

c) access is protected by default

d) none of the mentioned

**Answer: a**

1. **What will be the output of the code:**

#include<cstdio>

#include<iostream>

using namespace std;

class Base

{

public:

Base()

{

cout<<"Base class constructor called"<<endl;

}

void display()

{

cout<<"Base class display function called:"<<endl;

}

~Base()

{

cout<<"Base class destructor called"<<endl;

}

};

class Derived: public Base

{

public:

Derived()

{

cout<<"Derived class constructor called"<<endl;

}

void display()

{

cout<<"Derived class display function called:"<<endl;

}

~Derived()

{

cout<<"Derived class destructor called"<<endl;

}

};

int main()

{

Derived derived\_obj;

Base \*ptr\_to\_base\_obj=&derived\_obj;

ptr\_to\_base\_obj->display();

return 0;

}

**It will print like:**

**Base class constructor called**

**Derived class constructor called**

**Base class display function called:**

**Derived class destructor called**

**Base class destructor called**

**What will be the output of the following code:**#include<cstdio>

#include<iostream>

using namespace std;

class Base

{

public:

Base()

{

cout<<"Base class constructor called"<<endl;

}

void display()

{

cout<<"Base class display function called:"<<endl;

}

~Base()

{

cout<<"Base class destructor called"<<endl;

}

};

class Derived: public Base

{

public:

Derived()

{

cout<<"Derived class constructor called"<<endl;

}

void display()

{

cout<<"Derived class display function called:"<<endl;

}

~Derived()

{

cout<<"Derived class destructor called"<<endl;

}

};

int main()

{

Base \*ptr\_to\_base\_obj=new Derived();

ptr\_to\_base\_obj->display();

return 0;

}

**What will be the output of the following code?**

Base class constructor called

Derived class constructor called

Base class display function called:

**Now, can you see the difference?** Obviously.  
  
**But, can you tell me what causes the difference?**

In the first program, the pointer neither invokes constructor nor invokes destructor. The constructor and destructor logs are printed as the derived\_objis created and goes out of scope.   
  
In the second program, the derived class object is created dynamically. Thus, memory for it is allocated from heap. Now, since, derived class object is created first, Base class’s constructor is called and then derived class’s constructor is called. Now, can you see any destructor logs? Since, the memory is allocated from heap, it has not gone out of scope.