PPL QUIZ 2

u19cs012@coed.svnit.ac.in Switch account

Draft saved

Your email will be recorded when you submit this form

Quiz

Which one of the following is correct, when a class grants friend status to another class?

○ The member functions of the class generating friendship can access the members of the friend class.

● All member functions of the class granted friendship have unrestricted access to the members of the class granting the friendship.

○ Class friendship is reciprocal to each other.

○ There is no such concept.

Clear selection

Which of the following is true?

○ Static methods cannot be overloaded.

○ Static data members can only be accessed by static methods.

● Non-static data members can be accessed by static methods.

○ Static methods can only access static members (data and methods)

Clear selection

What is an abstract class?

○ Class declared with abstract keyword

○ Class which has exactly one virtual function

● Class which hash at least one pure virtual function

○ None of these

Clear selection

Which of the following statement is incorrect?

○ An object is an instance of the class

○ A friend function can access private members of a class

○ Members of the class are private by default

● None of these

Clear selection

Output ?

```
Predict the output of following C++ program
#include<iostream>
using namespace std;

class Base
{
public:
    virtual void show() { cout<<" In Base n"; }
};

class Derived: public Base
{
public:
    void show() { cout<<"In Derived n"; }
};

int main(void)
{
    Base *bp = new Derived;
    bp->Base::show();  // Note the use of scope resolution here
    return 0;
}
```

● In Base

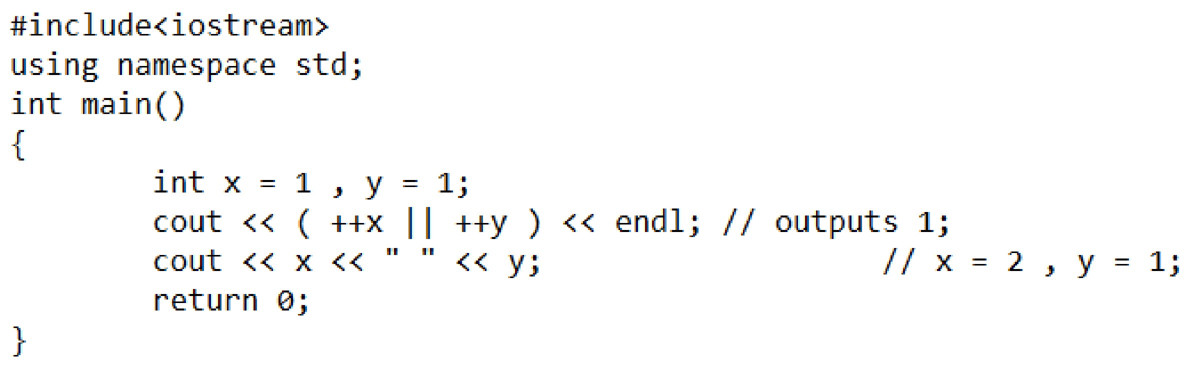
○ In Derived

○ Compiler Error

○ Runtime Error

Clear selection

New Line in options just represents its nature of output.



```
#include<iostream>
using namespace std;
int main()
{
        int x = 1 , y = 1;
        cout << ( ++x || ++y ) << endl; // outputs 1;
        cout << x << " " << y;                    // x = 2 , y = 1;
        return 0;
}
```

● 1 New Line - 2 1

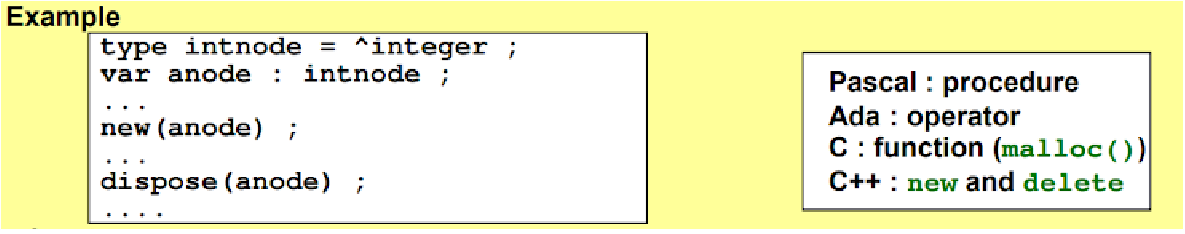
○ 1 New Line - 1 1

○ 1 New Line - 2 2

○ 1 New Line - 1 2

Clear selection

Following is an example of



Example

```
type intnode = ^integer ;
var anode : intnode ;
...
new(anode) ;
...
dispose(anode) ;
....
```

Pascal : procedure
Ada : operator
C : function (`malloc()`)
C++ : `new` and `delete`

○ Static Variables

○ Stack Dynamic Variables

● Implicit Heap Dynamic Variables

○ Explicit Heap Dynamic Variables

Doubt!

Clear selection

Output of following C++ program?

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

int main()
{
  int x = 10;
  int& ref = x;
  ref = 20;
  cout << "x = " << x << endl ;
  x = 30;
  cout << "ref = " << ref << endl;
  return 0;
}
```

● x = 20 ref = 30

○ x = 20 ref = 20

○ x = 10 ref = 30

○ x = 30 ref = 30

Clear selection

A member function can always access the data in _________.

● the class of which it is member

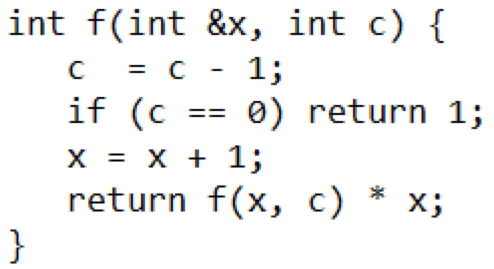
○ the object of which it is a member

○ the public part of its class

○ the private part of its class

Clear selection

What is the return value of f(p, p) if the value of p is initialized to 5 before the call? Note that the first parameter is passed by reference, whereas the second parameter is passed by value.



```
int f(int &x, int c) {
   c  = c - 1;
   if (c == 0) return 1;
   x = x + 1;
   return f(x, c) * x;
}
```

○ 3024

● 6561

○ 55440

○ 161051

Clear selection

Which of the following is true about templates. 1) Template is a feature of C++ that allows us to write one code for different data types. 2) We can write one function that can be used for all data types including user defined types. Like sort(), max(), min(), ..etc. 3) We can write one class or struct that can be used for all data types including user defined types. Like Linked List, Stack, Queue ..etc. 4) Template is an example of run time polymorphism.

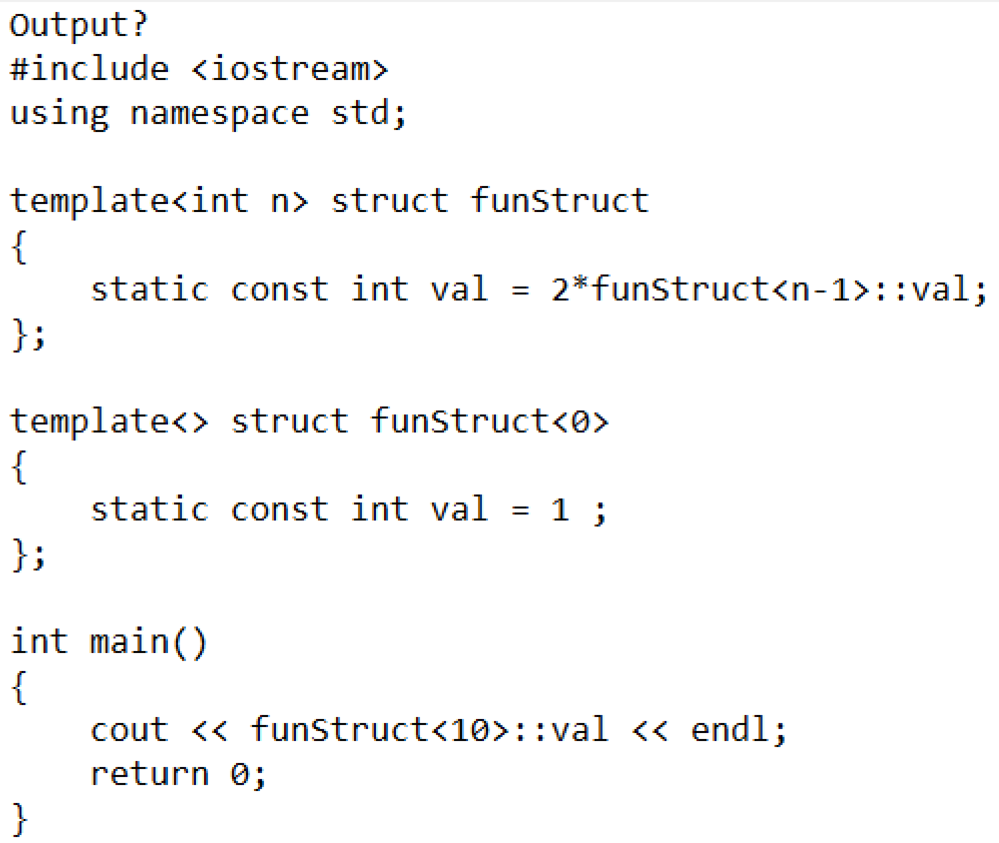
○ 1 and 2

● 1, 2 and 3

○ 1, 2 and 4

○ 1, 2, 3 and 4

Clear selection



```
Output?
#include <iostream>
using namespace std;

template<int n> struct funStruct
{
    static const int val = 2*funStruct<n-1>::val;
};

template<> struct funStruct<0>
{
    static const int val = 1 ;
};

int main()
{
    cout << funStruct<10>::val << endl;
    return 0;
}
```

○ Compiler Error

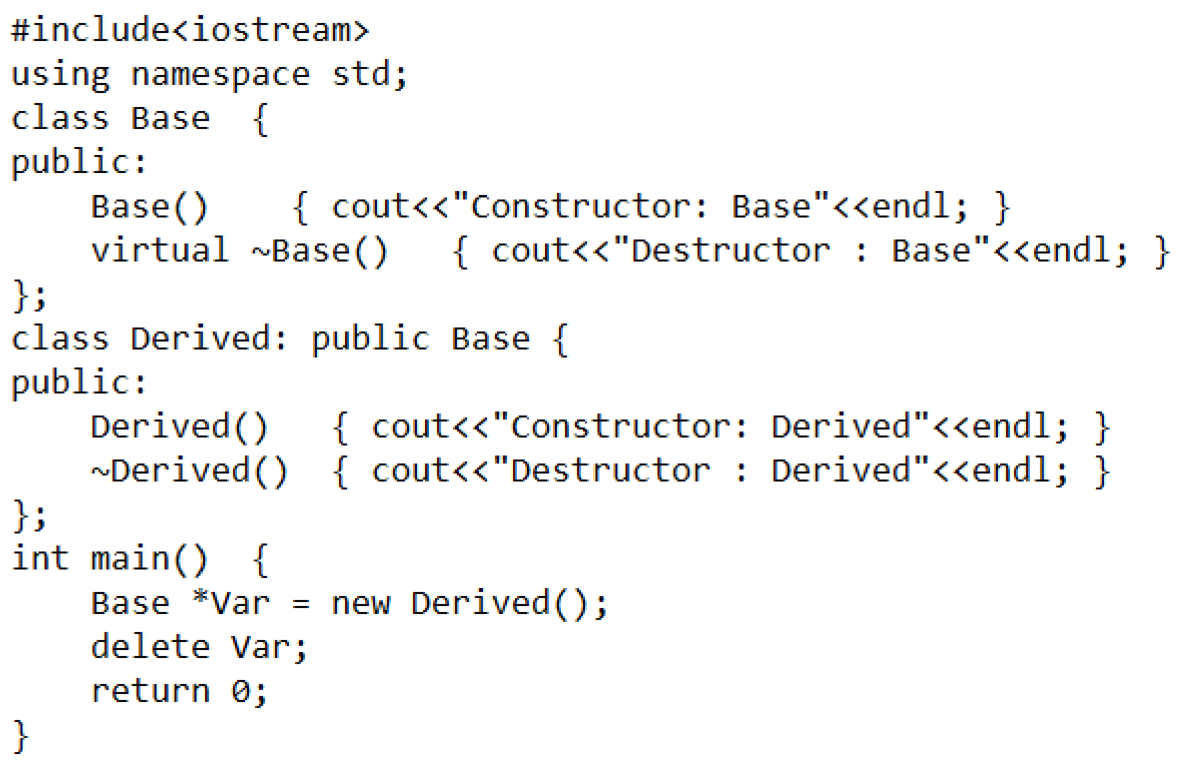
● 1024

○ 2

○ 1

Clear selection

Output?



```
#include<iostream>
using namespace std;
class Base  {
public:
    Base()    { cout<<"Constructor: Base"<<endl; }
    virtual ~Base()   { cout<<"Destructor : Base"<<endl; }
};
class Derived: public Base {
public:
    Derived()   { cout<<"Constructor: Derived"<<endl; }
    ~Derived()  { cout<<"Destructor : Derived"<<endl; }
};
int main()  {
    Base *Var = new Derived();
    delete Var;
    return 0;
}
```

○ Constructor: Base Constructor: Derived Destructor : Derived Destructor : Base

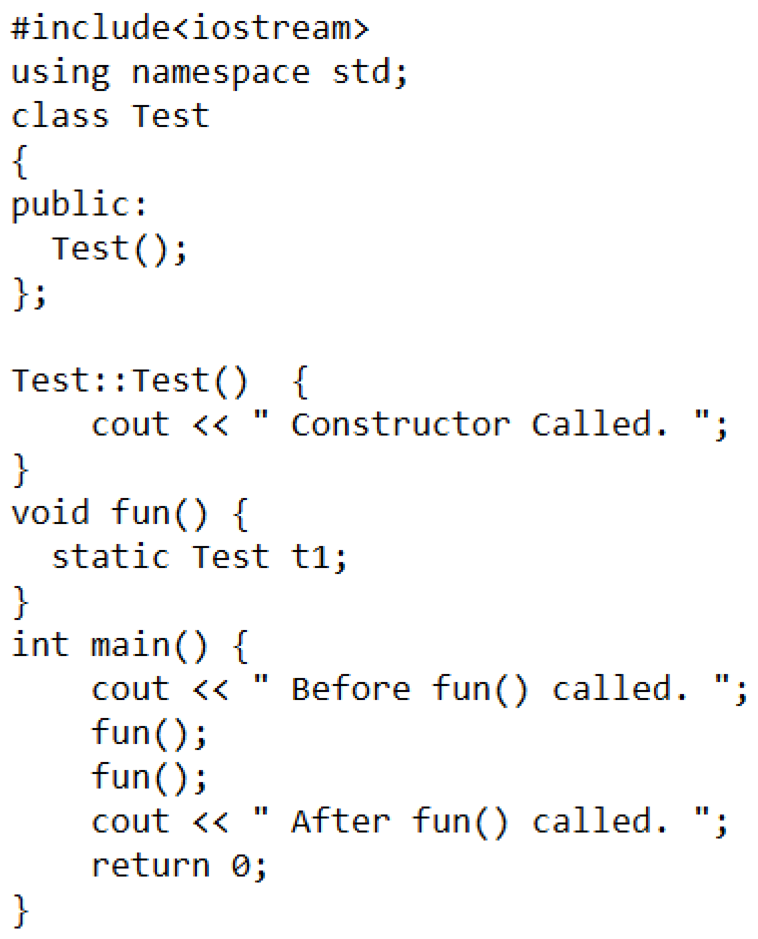
● Constructor: Base Constructor: Derived Destructor : Base

○ Constructor: Base Constructor: Derived Destructor : Derived

○ Constructor: Derived Destructor : Derived Destructor : Base

Clear selection

Output?



```
#include<iostream>
using namespace std;
class Test
{
public:
  Test();
};

Test::Test()  {
    cout << " Constructor Called. ";
}
void fun() {
  static Test t1;
}
int main() {
    cout << " Before fun() called. ";
    fun();
    fun();
    cout << " After fun() called. ";
    return 0;
}
```

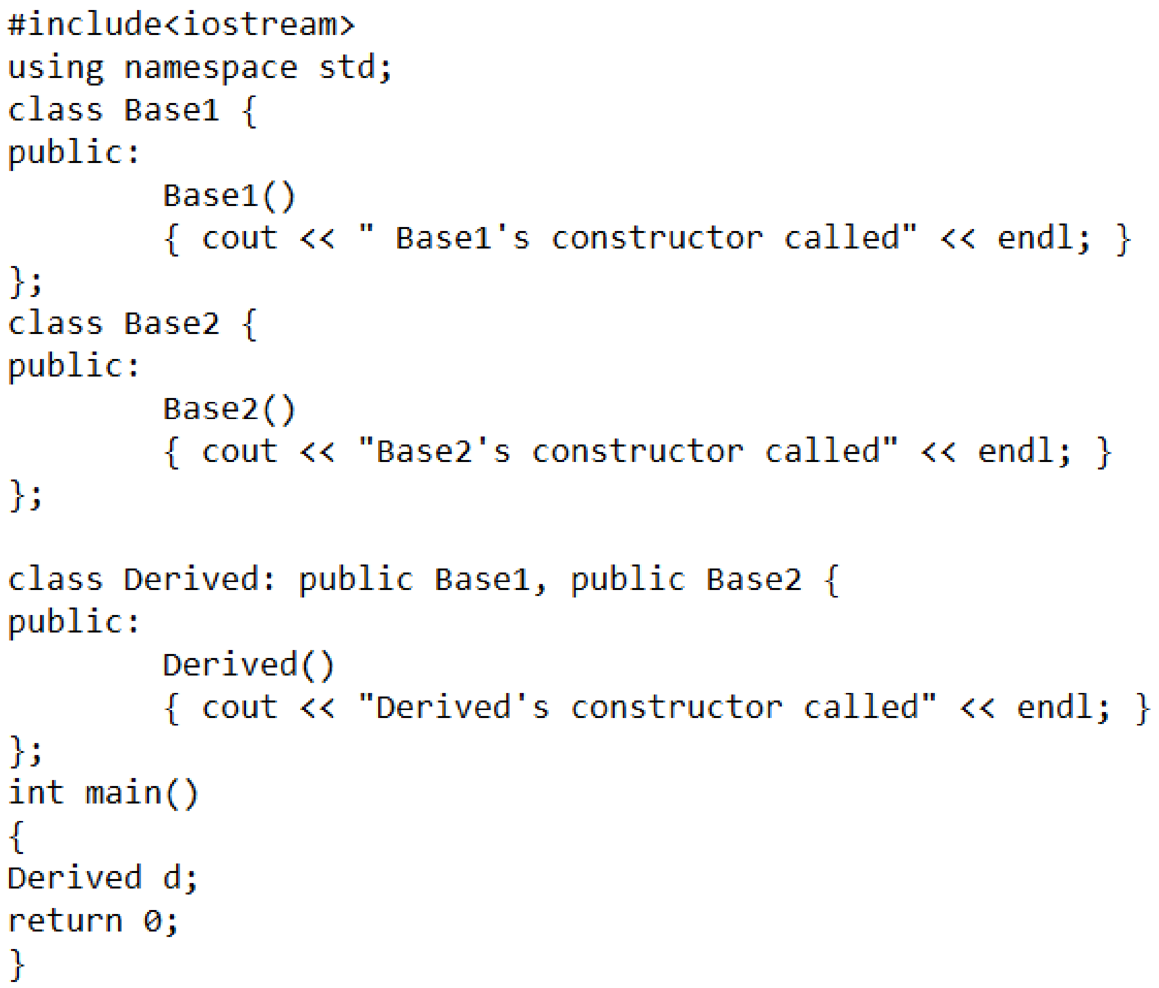
○ Constructor Called. Before fun() called. After fun() called.

○ Before fun() called. Constructor Called. Constructor Called. After fun() called.

● Before fun() called. Constructor Called. After fun() called.

○ Constructor Called. Constructor Called. After fun() called.Before fun() called.

Clear selection



```
#include<iostream>
using namespace std;
class Base1 {
public:
        Base1()
        { cout << " Base1's constructor called" << endl; }
};
class Base2 {
public:
        Base2()
        { cout << "Base2's constructor called" << endl; }
};

class Derived: public Base1, public Base2 {
public:
        Derived()
        { cout << "Derived's constructor called" << endl; }
};
int main()
{
Derived d;
return 0;
}
```

● Base1’s constructor called Base2’s constructor called Derived’s constructor called

○ Base1’s constructor called Derived’s constructor called Base2’s constructor called

○ Derived’s constructor called Base1’s constructor called Base2’s constructor called

○ Derived’s constructor called Base1’s constructor called

Clear selection

How to create a dynamic array of pointers (to integers) of size 10 using new in C++?

○ int *arr = new int *[10];

● int **arr = new int *[10];

○ int *arr = new int [10];

○ Not Possible

Clear selection

Which of the following is true about constructors. 1) They cannot be virtual. 2) They cannot be private. 3) They are automatically called by new operator.

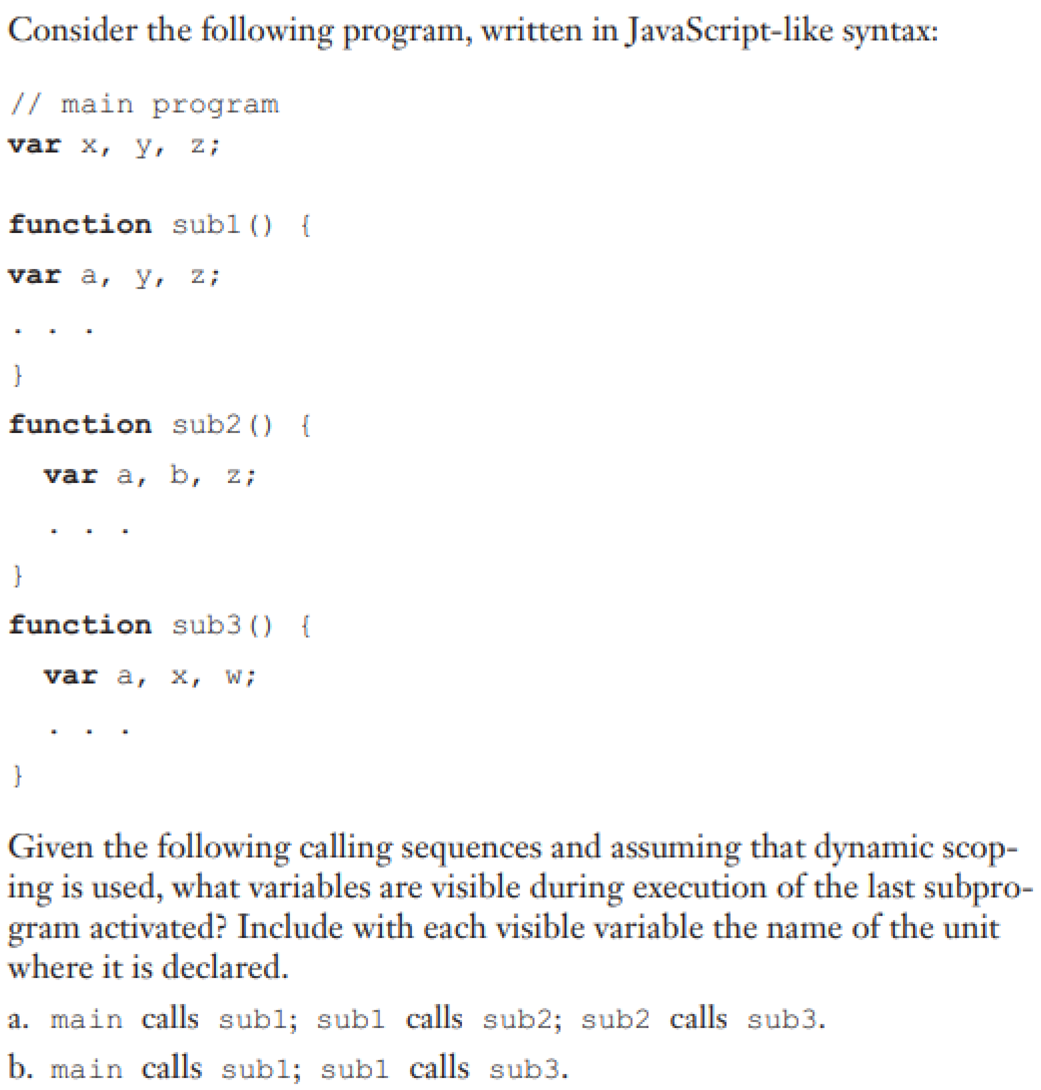
● All 1, 2, and 3

○ Only 1 and 3

○ Only 1 and 2

○ Only 2 and 3

Clear selection



Consider the following program, written in JavaScript-like syntax:

```
// main program
var x, y, z;

function sub1() {
var a, y, z;
. . .
}
function sub2() {
  var a, b, z;
  . . .
}
function sub3() {
  var a, x, w;
  . . .
}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last subprogram activated? Include with each visible variable the name of the unit where it is declared.

a. main calls sub1; sub1 calls sub2; sub2 calls sub3.

b. main calls sub1; sub1 calls sub3.

○ main calls sub1; sub1 calls sub2; sub2 calls sub3. Answer: a x in sub3, sub1 calls sub2, sub2 calls sub3 main calls sub1; sub1 calls sub3. Answer: a x q in sub3, y z in sub1

○ main calls sub1; sub1 calls sub2; sub2 calls sub3. Answer: a x w in sub3, sub1 calls sub2, sub2 calls sub3 main calls sub1; sub1 calls sub3. Answer: a x q in sub3, y z in sub1

○ main calls sub1; sub1 calls sub2; sub2 calls sub3. Answer: a x w in sub3, sub1 calls sub2, sub2 calls sub3 main calls sub1; sub1 calls sub3. Answer: a q in sub3, y z in sub1

● main calls sub1; sub1 calls sub2; sub2 calls sub3. Answer: a x w in sub3, sub1 calls sub2, sub2 calls sub3 main calls sub1; sub1 calls sub3. Answer: a x q in sub3, y x z in sub1

Clear selection

Which of the following is true about new when compared with malloc. 1) new is an operator, malloc is a function 2) new calls constructor, malloc doesn't 3) new returns appropriate pointer, malloc returns void * and pointer needs to typecast to appropriate type.

○ 1 and 3

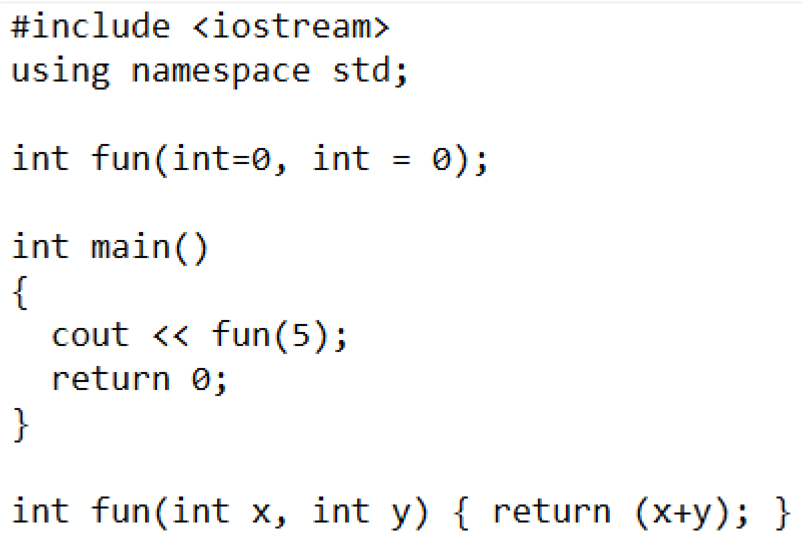
○ 2 and 3

○ 1 and 2

● All of these

Clear selection

Output of following program?



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

int fun(int=0, int = 0);

int main()
{
   cout << fun(5);
   return 0;
}

int fun(int x, int y) { return (x+y); }
```

○ Compiler Error

● 5

○ 0

○ 10

Clear selection

If a function is friend of a class, which one of the following is wrong?

○ A function can only be declared a friend by a class itself.

○ Friend functions are not members of a class, they are associated with it.

● Friend functions are members of a class.

○ It can have access to all members of the class, even private ones.

Clear selection

```
Output of following program
#include<iostream>
using namespace std;

class Base {};
class Derived: public Base {};
int main()
{
   Derived d;
   try {
       throw d;
   }
   catch(Base b) {
        cout<<"Caught Base Exception";
   }
   catch(Derived d) {
        cout<<"Caught Derived Exception";
   }
   return 0;
}
```

○ Caught Derived Exception

● Caught Base Exception

○ Compiler Error

○ None of these

Clear selection

Which of the following operator functions cannot be global, i.e., must be a member function.

○ new

○ delete

○ Conversion Operator

● All of these

Clear selection

Assume that an integer takes 4 bytes and there is no alignment in following classes, predict the output.

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

class base {
    int arr[10];
};

class b1: public base { };

class b2: public base { };

class derived: public b1, public b2 {};

int main(void)
{
  cout << sizeof(derived);
  return 0;
}
```

○ 40

● 80

○ 0

○ 4

Clear selection

Assume the following JavaScript program was interpreted using static-scoping rules. What value of x is displayed in function sub1? Under dynamic-scoping rules, what value of x is displayed in function sub1?

```
var x;
function sub1() {
   document.write("x = " + x + "<br />");
}
function sub2() {
   var x;
   x = 10;
   sub1();
}
x = 5;
sub2();
```

○ Static scoping: x is 10 Dynamic scoping: x is 5

● Static scoping: x is 5 Dynamic scoping: x is 10

○ Static scoping: x is 5 Dynamic scoping: x is 5

○ Static scoping: x is 10 Dynamic scoping: x is 10

Clear selection

If a base class and derived class each include a member function with the same name. Function from which class will be called if called by an object of the derived class

○ Member function of the base class

○ Member function of the derived class

● Depend on the parameter

○ None of these

Clear selection

Which of the followings is/are automatically added to every class, if we do not write our own.

○ Copy Constructor

○ Assignment Operator

○ A constructor without any parameter

● All of these

Clear selection

The associativity of which of the following operators is Left to Right, in C++ ?

● Unary Operator

○ Logical not

○ Array element access

○ addressof

Clear selection

Which of the following is true about this pointer?

○ It is passed as a hidden argument to all function calls

● It is passed as a hidden argument to all non-static function calls

○ It is passed as a hidden argument to all static functions

○ None of these

Clear selection

Consider the below C++ program.

```
#include<iostream>
using namespace std;
class A
{
public:
     A(){ cout <<"1";}
     A(const A &obj){ cout <<"2";}
};

class B: virtual A
{
public:
    B(){cout <<"3";}
    B(const B & obj){cout<<"4";}
};

class C: virtual A
{
public:
   C(){cout<<"5";}
   C(const C & obj){cout <<"6";}
};

class D:B,C
{
public:
    D(){cout<<"7";}
    D(const D & obj){cout <<"8";}
};

int main()
{
   D d1;
   D d(d1);
}
```

○ 2

○ 4

○ 6

● All of these

Back Submit Clear form