1. What is the correct syntax for lambda expression in C++11?
   1. [capture](parameters)->return-type {body}
   2. [parameters](capture)->return-type {body}
   3. [capture][parameters]->return-type {body}
   4. (capture)(parameters)->return-type {body}
2. Does below code compiles successfully on C++11?

class C

{

int a=7;

public:

C(){}

};

* 1. Yes
  2. No

1. What do you understand from lexical analysis?
   1. Checking the Grammar
   2. Finding the type of names and expressions
   3. Making up tokens from characters
   4. None of these
2. In C++11 which data type is newly introduced among these?
   1. Long
   2. long int
   3. longs long
   4. long long
3. Which keyword in C++ 11 is used to prevent a virtual function from being overridden?
   1. Final
   2. Static
   3. Stop
   4. Friend
4. Which flag is used for call\_once in c++?(
   1. once\_flag
   2. flag\_once
   3. flaged
   4. called\_flag
5. Which one maintains a reference count in c++?
   1. Shared\_ptr
   2. Unique\_ptr
   3. Const\_ptr
   4. Weak\_ptr
6. Look at this code

int x = 42;

    std::thread t(increment, x);

* 1. X is passed by value
  2. X is passed by refrence
  3. Compile Error
  4. Runtime Error

1. Which method is used to start a thread pool (ThreadPool pool)?
   1. Pool.init()
   2. Pool.start()
   3. Pool.create()
   4. Pool.submit()
2. Which technique is used to invoke RValue Reference?
   1. Std::move
   2. Std::forward
   3. Std::perfect
   4. Std::ref
3. What object does Std::async returns ?
   1. Promise
   2. Future
   3. Handle
   4. Shared\_Promise
4. Which return type of a function deduces a non-reference or reference type depending on return value?
   1. Auto
   2. Auto&
   3. Decltype(auto)
   4. Decltype(auto&)
5. The output of the program is

void func(int &r) { r++; }

template<class F, class P> void g(F f, P t) { f(t); }

void main()

{

    int i = 10;

    g(func, i);

    std::cout << i;

    g(func, std::ref(i));

    std::cout << i;

}

* 1. 10 10
  2. 10 11
  3. 11 11
  4. 11 10

1. The output of the program is   
    void Hello(int \*pt)

{

    cout << "Apple ";

}

void Hello(int p)

{

    cout << "Orange " ;

}

void main()

{

    Hello(NULL);

    Hello(nullptr);

}

* 1. Apple Orange
  2. Apple Apple
  3. Orange Orange
  4. Orange Apple

1. When we use generic lambda   
    auto l0([](auto x){

         return x \* 2;

     });

    static\_assert(std::is\_same<decltype(l0(1)), int>(), "");//1

    static\_assert(std::is\_same<decltype(l0(1.f)), float>(), "");//2

    static\_assert(std::is\_same<decltype(l0(1.)), double>(), "");//3

* 1. All 1,2,3 asserts passes
  2. Only 1 and 2 passes
  3. Only 1 and 3 passes
  4. Only 2 and 3 passes

1. Which of the following is a correct syntax to pass a Function Pointer as an argument?
   1. void pass(int (\*fptr)(int, float, char)){}
   2. void pass(\*fptr(int, float, char)){}
   3. void pass(int (\*fptr)){}
   4. void pass(\*fptr){}
2. The Member of a class which can be manipulated even if it is accessed via const object or Inspectors.
   1. Instance member
   2. Const member
   3. Mutable member
   4. Refrence member
3. What is the output of the program   
    struct CA

{

void fun() { cout<<"Apple"<<endl; }

};

struct Smart

{

CA \*pt;

Smart():pt(new CA()){}

CA \* operator->(){ return pt; }

void fun() {cout<<"Orange "<<endl; }

};

void main()

{

Smart \*p=new Smart();

p->fun();

}

* 1. Apple Orange
  2. Apple
  3. Orange
  4. Orange Apple

1. Which Convention is used to stack unwind in callee?
   1. \_\_stdcall
   2. \_\_thiscall
   3. \_\_cdecl
   4. \_\_fastcall
2. What is used for a function to execute at compile time
   1. Decltype
   2. Constexpr
   3. Lambda
   4. functor