TERM	COURSE NAME	COURSE CODE	VERSION
Fall-2019-Quiz-9	Object-Oriented Software Development using C++	OOP345	**A**

Code1.0

```
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;
 int main() {
     char s[] = "A C string";
     std::cout << std::hex;
1.   for (int i = 0; s[i]; i++)
2.   {
3.       std::cout << (int*)&s[i]<< " : ";
4.       std::cout  << s[i] << std::endl;
5.   }
6.   for (int i = 0; s[i]; i++){
7.        std::cout << (int*)&s[i] << " : ";
8.        std::cout << &s[i] << std::endl;
9.   }
 }
```

Answer Questions 1-5 using Code 1.0

1. The first iteration of line 4 prints
   a. A C string
   b. A
   c. C
   d. C string
   e. tring
2. The third iteration of line 4 prints
   a. A C string
   b. A
   c. C
   d. C string
   e. tring
3. The first iteration of line 8 prints
   a. A C string
   b. A
   c. C
   d. C string
   e. tring
4. The third iteration of line 8 prints
   a. A C string
   b. A
   c. C

    d. C string
    e. tring
5. The sixth iteration of line 8 prints
    a. A C string
    b. A
    c. C
    d. C string
    e. tring

Code2.0

```
// Pointing to a String Literal
 // ptrToStringLiteral.cpp

 #include <iostream>

 int main() {
     char *p  = "Avoid overwriting";

     p[0] = 'a';
     std::cout << p << std::endl;
 }
```

Answer Questions 6 using Code 2.0

6. Code 2.0 will result in
    a. Compile time error
    b. Run-time error
    c. All of the above
    d. None of the above

Code3.0

```
// String Constants
 // ptrToConstStringLiteral.cpp

 #include <iostream>

 int main() {
     const char *p  = "Avoid overwriting"; // good coding style

     p[0] = 'a';
     std::cout << p << std::endl;
 }
```

Answer Questions 7 using Code 3.0

7. Code 3.0 will result in
    a. Compile time error
    b. Run-time error
    c. All of the above
    d. None of the above

Code4.0

```
#include <iostream>
 int main() {
     double a[] = {1.1, 2.2, 3.3, 4.4 , 5.5};
     int i = 2;
     double* p;
     p = &a[1];
     std::cout << std::hex;
     1.   std::cout << a[0]     << " : ";
     2.   std::cout  << a     << std::endl;
     3.   std::cout << *(p + i) << " : ";
     4.   std::cout << p + i << std::endl;
     5.   std::cout << p[i]     << " : ";
     6.   std::cout << &p[i] << std::endl;
 }
```

Answer Questions 8-13 using Code 4.0

8. Line 1 will print
    a. 1.1
    b. The address of the first item in array **a**
    c. 4.4
    d. None of the above
9. Line 2 will print
    a. 1.1
    b. The address of the first item in array **a**
    c. 4.4
    d. None of the above
10. Line 3 will print
    a. 1.1
    b. The address of the fourth item in array **a**
    c. 4.4
    d. None of the above
11. Line 4 will print
    a. 1.1
    b. The address of the fourth item in array **a**
    c. 4.4
    d. None of the above
12. Line 5 will print
    a. 1.1
    b. The address of the fourth item in array **a**
    c. 4.4
    d. None of the above
13. Line 6 will print
    a. 1.1
    b. The address of the fourth item in array **a**
    c. 4.4
    d. None of the above

Code5.0

```
// Reference to a Pointer
 // ref_to_ptr.cpp
 #include <iostream>
 void swap(int*& a, int*& b) {
     int* t = a;
     a = b;
     b = t;
 }
 int main() {
     int x, y;
     int* p = &x;
     int* q = &y;
1.   std::cout << "p = " << p << std::endl;
2.   std::cout << "q = " << q << std::endl;
3.   swap(p, q);
4.   std::cout << "p = " << p << std::endl;
5.   std::cout << "q = " << q << std::endl;
 }
```

Answer Questions 14-15 using Code 5.0

14. Line 1 will print the same value as
    a. Line 2
    b. Line 4
    c. Line 5
    d. None of the above

15. Line 2 will print the same value as
    a. Line 1
    b. Line 4
    c. Line 5
    d. None of the above

Code6.0

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

int main()
{
  const char* str1   = "Hello";
  const char  str2[] = "Hello";

1.  cout << "Test 1: " << (str1 == "Hello" ? "same" : "different"). << endl;
2.  cout << "Test 2: " << (str2 == "Hello" ? "same" : "different") << endl;

}
```

Answer Questions 16-17 using Code 6.0

16. Line 1 print:
    a. Test 1: same
    b. Test 2: different
    c. All of the above
    d. None of the above
17. Line 2 print:
    a. Test 1: same
    b. Test 2: different
    c. All of the above
    d. None of the above

Code7.0

```
#include <iostream>
 class Title {
     const char* title;
     const char* validTitle() const {
         if (!title[0]) throw "invalid title";
         return title;
     }
   public:
     Title(const char* t) : title(t) {}
     void display() const {
         std::cout << validTitle() << std::endl;
     }
 };
 // ********************************************
 void display(const char* t) {
     Title* tt = new Title(t);
     tt->display(); // may throw an exception!
     delete tt;
     std::cout << "Dynamic memory deleted" << std::endl;
 }
 int main() {
     const char* s[] = {"Mr.", "Ms.", "", "Dr."};
    1.   for (auto x : s) {
    2.       try {
    3.           ::display(x);
    4.       } catch(const char* msg) {
    5.             std::cerr << msg << std::endl;
    6.             std::cerr << "Memory Leak " << std::endl;
    7.   }
     }
 }
```

Answer Questions 18-19 using Code 7.0

18. The first iteration of line 3 will trigger the printing of the message:
    a. Dynamic memory deleted
    b. Memory Leak
    c. All of the above
    d. None of the above
19. The third iteration of line 3 will trigger the printing of the message:
    a. Dynamic memory deleted
    b. Memory Leak
    c. All of the above
    d. None of the above

Code8.0

```
#include <iostream>
 #include <memory>

 class Title {
     const char* title;
     const char* validTitle() const {
         if (!title[0]) throw "invalid title";
         return title;
     }
   public:
     Title(const char* t) : title(t) {}
     void display() const {
         std::cout << validTitle() << std::endl;
     }
 };
 // ******************************************

 void display(const char* t) {
     std::unique_ptr<Title> tt(new Title(t));
     tt->display();
     Title ttt = *tt;
     ttt.display();
 }

 int main() {
     const char* s[] = {"Mr.", "Ms.", "", "Dr."};

     for (auto x : s) {
         try {
             display(x);
         } catch(const char* msg) {
             std::cerr << msg << std::endl;
         }
     }
 }
```

Answer Questions 20 using Code 8.0

20. Will this code trigger a memory leak :
    a. Yes
    b. NO