TERM	COURSE NAME	COURSE CODE	VERSION
Winter-2020-Quiz-2	Object-Oriented Software Development using C++	OOP345	**A**

Code.10

```
1. unsigned char x = 0;
2. unsigned char y = 150;
3. std::cout << " Entering the loop " << std::endl;
4. for ( ; x < 2*y; x++ )
5. {
6.      std::cout << " x = " << (int) x <<  std::endl;
7. }
8. std::cout << " Came out of the loop" << std::endl;
9. std::cout << " x = " << (int) x <<  std::endl;
```

1. Code 1.0 - Line 9 code 1.0 will print:
    a. 300
    b. 301
    c. 299
    d. None of the above
2. Code 1.0 – Line 4 code 1.0 will loop through this many iterations:
    a. 300
    b. 301
    c. 299
    d. None of the above

Code2.0

```
1. int n0 = 7;
2. int n1 = 7.2;
3. int n2 {6};
4. int n3 = {5.5}; // = is redundant

5. std::cout << "n0 = " << n0 << std::endl;
6. std::cout << "n1 = " << n1 << std::endl;
7. std::cout << "n2 = " << n2 << std::endl;
```

3. Code 2.0 will compile successfully
    a. YES
    b. NO
4. Code 2.0, assuming any compilation error are fixed, line 5 will output
    a. 7
    b. 7.1
    c. 6
    d. 5.5

5. Code 2.0, assuming any compilation error are fixed, line 6 will output
   a. 7
   b. 7.1
   c. 6
   d. 5.5
6. Code 2.0, assuming any compilation error are fixed, line 7 will output
   a. 7
   b. 7.1
   c. 6
   d. 5.5
7. A C++17 compiler can infer the type of an object from a previously declared object.
   a. YES
   b. NO
8. The keyword **auto** specifies inference
   a. YES
   b. NO

code 3.0

```
1. int a[] {1, 2, 3, 4, 5, 6};
2. const auto n = 6;

3. for (auto i = 0; i < n; i++)
4.     std::cout << a[i] << ' ';
5. std::cout << std::endl;
```

9. Code 3.0, the first iteration of line 4 is
   a. 1
   b. 2
   c. 3
   d. 4
   e. 5
   f. 6
10. Code 3.0, the second iteration of line 4 is
   a. 1
   b. 2
   c. 3
   d. 4
   e. 5
   f. 6
11. Code 3.0, the third iteration of line 4 is
   a. 1
   b. 2
   c. 3
   d. 4
   e. 5
   f. 6
12. Any attempt to dereference a pointer that holds the value **nullptr** causes a run-time error.
   a. NO
   b. YES
13. A wild pointer is a pointer that has been initialized to an address.
   a. NO
   b. YES
14. It is good style to initialize every wild pointer to **nullptr**:
   a. YES
   b. NO

15. A generic pointer type is a pointer type that is not associated with any particular type.
    a. NO
    b. YES
16. The keyword **void** identifies a generic pointer type

```
void* p; // generic pointer type
```

    a. YES
    b. NO
17. Which code snippet will cause compile errors:

Code 4.0

```
int* i;
char* c;
i = c; //
      //
```

Code 5.0

```
int* i;
char* c;
i = static_cast<int*>(
    static_cast<void*>(c));
```

    a. Code 5.0
    b. Code 4.0
    c. All of the above
    d. None of the above
18. Which of the two codes will cause compile errors:

Code 6.0

```
int i = 5;
void* v = &i;
std::cout << *v << std::endl;
```

Code 7.0

```
int i = 5;
void* v = &i;
std::cout << *static_cast<int*>(v) << std::endl;
```

    a. Code 6.0
    b. Code 7.0
    c. All of the above
    d. None of the above

code 8.0

```
template <typename T>
void print(T& val)
{
     std::cout << "l-value: " << val << std::endl;
}
template <typename T>
void print(T&& val)
{
     std::cout << "r-value: " << val << std::endl;
}

int main() {

   1. static int xyz = 55;
   2. int a{900};
   3. float c(30);
   4. print(a);
   5. print(float(30));
   6. print( a + c );
   7. print(xyz);
   8. print(std::move(a));
}
```

19. Code 8.0, line **4** will print.
    a. r-value 900
    b. l-value 900
    c. All of the above
    d. None of the above
20. Code 8.0, line **5** will print.
    a. r-value 30
    b. l-value 30
    c. All of the above
    d. None of the above
21. Code 8.0, line **6** will print.
    a. r-value 930
    b. l-value 930
    c. All of the above
    d. None of the above
22. Code 8.0, line **7** will print.
    a. r-value 55
    b. l-value 55
    c. All of the above
    d. None of the above
23. Code 8.0, line **8** will print.
    a. r-value 900
    b. l-value 900
    c. All of the above
    d. None of the above

Code 9.0

```
1. int foo (10);
2. auto bar = std::ref(foo);
3. ++bar;
4. ++foo
5. std::cout << foo << '\n';
```

24. Code 9.0, line **5** will print.
    a. 10
    b. 12
    c. 11
    d. All of the above
    e. None of the above

Code 10

```
1. int foo (10);
2. int bar;
3. bar = std::ref(foo);
4. ++bar;
5. std::cout << foo << '\n';
6. std::cout << bar << '\n';
```

25. Code 10, line **5** will print.
    a. 10
    b. 12
    c. 11
    d. All of the above
    e. None of the above

26. Code 10, line **6** will print.
    a. 10
    b. 12
    c. 11
    d. All of the above
    e. None of the above

Code 11.0

```
1.   int a[]{1, 2, 3, 4, 5, 6};
2.   for (auto e : a){
3.       e += 2;
4.   }
5.   for (auto& e : a){
6.       e++;
7.   }
8.   for (auto& e : a){
9.       std::cout << e << ' ';
10.  }
11.  std::cout << std::endl;
```

27. Code 11, the first iteration of line **9** will print.
    a. 4
    b. 1
    c. 2
    d. 3
    e. None of the above
28. Code 11, the second iteration of line **9** will print.
    a. 4
    b. 1
    c. 2
    d. 3
    e. None of the above
29. Code 11, the third iteration of line **9** will print.
    a. 3
    b. 5
    c. 4
    d. 6
    e. None of the above
30. Code 11, the fourth iteration of line **9** will print.
    a. 3
    b. 5
    c. 4
    d. 6
    e. None of the above
31. Which of the following statements are correct?
    a. Classes are strongly encapsulated.
    b. The members of a **class** are private by default, which facilitates the hiding of their information.
    c. All of the above
    d. None of the above
32. Which of the following statements are correct?
    a. Structures and unions are weakly encapsulated.
    b. The members of a **struct** or **union** are public by default, which facilitates the sharing of their information.
    c. All of the above
    d. None of the above

Code 12.0

```
class Subject {
     unsigned number;
     char desc[41];
     Subject preRequisite;
 };
```

33. Code 12.0 will compile?
    a. YES
    b. NO

Code 13.0

```
1. class Subject{
2.      const int id = 100;
3.      Subject(): id(5){
4.            id = 5;
5.       }
6. };
```

34. Code 13.0, which line will cause compilation error
    a. 2
    b. 3
    c. 4
    d. All of the above
    e. None of the above

Code 14.0

```
void func_ranges0(){
    unsigned char x = 0;
    unsigned char y = 150;
    x = 2*y;
     std::cout << " x = " << (int) x <<  std::endl;
}
```

35. Code 14.0, will output:
    a. x = 44
    b. x = 300
    c. None of the above

Also Cover the following in the lecture:

- Copy & Move Constructor
- Copy & Move Assignment
- Class-Variable & Class-Functions