| TERM | COURSE NAME | COURSE CODE | VERSION |
|---|---|---|---|
| Fall-2019-Quiz-Lecture-4 | Object-Oriented Software Development using C++ | OOP345 | **A** |

**Code1.0**

```
// Exception Handling
// exceptions.cpp

#include <iostream>

void divide(double a[], int i, int n, double divisor) {
    if(i < 0 || i >= n)
       throw "Outside bounds";
    else if(i == n / 2)
      throw i;
    else if(divisor == 0)
      throw divisor;
    else
       a[i] = i / divisor;
}

int main() {
   bool keepdividing = true;
   double a[] = {1.1,2.2,3.3,4.4,5.5,6.6}, divisor;
   int i, n = sizeof a / sizeof a[0];

   do {
      try {
         std::cout << "Index: ";
         std::cin >> i;
         std::cout << "Divisor: ";
         std::cin >> divisor;
         divide(a, i, n, divisor);
         std::cout << "a[i] = " << a[i] << std::endl;
         std::cout << "Continuing ..." << std::endl;
      }
      catch(const char* msg) {
         std::cout << msg << std::endl;
         keepdividing = false;
      }
      catch(int& value)
      {
         std::cout << "Index is " << value << std::endl;
         std::cout << "a[i] = " << a[i] << std::endl;
         std::cout << "Continuing ..." << std::endl;
      }
      catch(...) {
         std::cout << "Zero Division!" << std::endl;
         std::cout << "a[i] = " << a[i] << std::endl;
         std::cout << "Continuing ..." << std::endl;
      }
   } while (keepdividing);
}
```

1. Code 1.0, If Index = 45, Divisor = 3, the following catch block will be executed
   a. catch(const char* msg)
   b. catch(int& value)
   c. catch(...)
   d. All of the above
   e. None of the above

2. Code 1.0, If Index = 3, Divisor = 2, the following catch block will be executed
   a. catch(const char* msg)
   b. catch(int& value)
   c. catch(...)
   d. All of the above
   e. None of the above
3. Code 1.0, If Index = 5, Divisor = 0, the following catch block will be executed
   a. catch(const char* msg)
   b. catch(int& value)
   c. catch(...)
   d. All of the above
   e. None of the above

**Code2.0**

```
1.  // No Exceptions - compile on GCC
2.  // noexceptions.cpp

3.  #include <iostream>

4.  void d() { throw "d() throws\n"; }
5.  void e()
6.  {
7.  try { d(); }
8.  catch(const char* msg) { std::cout << msg; }
9.  }
10. void f() { throw "f() throws\n"; }
11. void g() noexcept { e(); }
12. void h() noexcept { f(); }

13. int main() {
14. std::cout << "Calling g: ";
15. g();
16. std::cout << "Calling h: ";
17. h();
18. std::cout << "Normal exit\n";
19. }
```

4. Code 2.0, Line 15 will result in:
   a. Triggers the code to terminate with uncaught exception
   b. Prints "Calling g: d() throws"
   c. Prints " Calling h: f() throws"
   d. All of the above
   e. None of the above

5. Code 2.0, Line 17 will result in:
   a. Triggers the code to terminate with uncaught exception
   b. Prints "Calling g: d() throws"
   c. Prints " Calling h: f() throws"
   d. All of the above
   e. None of the above

**Code3.0**

```
1.  #include <vector>
2.  #include <iostream>

3.  int main() {
4.      std::vector<double> prices;
5.      if(prices.empty())
6.         std::cout << "prices is empty" << std::endl;
7.      prices.push_back(10.43);
8.      prices.push_back(20.54);
9.      prices.push_back(32.43);
10.     for(int i = 0; i < prices.size(); i++)
11.         std::cout << prices[i] << "  ";
12.     std::cout << std::endl;
13.     prices.front() = 54.11;
14.     prices.pop_back();
15.     for(int i = 0; i < prices.size(); i++)
16.        std::cout << prices[i] << "  ";
17.     std::cout << std::endl;
18. }
```

6. Code 3.0, in Line 5, prices.empty() will return :
   a. True
   b. False
   c. All of the above
   d. None of the above
7. Code 3.0, Line 7 will print:
   a. 10.43
   b. 20.54
   c. 32.43
   d. None of the above
8. Code 3.0, Line 8 will print:
   a. 10.43
   b. 20.54
   c. 32.43
   d. None of the above
9. Code 3.0, Line 9 will print:
   a. 10.43
   b. 20.54
   c. 32.43
   d. None of the above
10. Code 3.0, Line 11 will print
    a. 10.43 20.54 32.43
    b. 32.43 20.54 10.43
    c. 20.31 15.64 10.5 32.43
    d. 32.43 20.31 15.64 10.5
    e. None of the above
11. Code 3.0, Line 16 will result in:
    a. 10.5 32.43
    b. 32.43 10.5
    c. 20.31 15.64 10.5 32.43
    d. 54.11 20.54
    e. None of the above

**<u>Code4.0</u>**

```
1.  // Iterators - Vectors
2.  // iterator.cpp

3.  #include <vector>
4.  #include <iostream>

5.  int main() {
6.      std::vector<double> prices;  // initially empty
7.      std::vector<double>::iterator i;

8.      prices.push_back(10.43); // add 10.43
9.      prices.push_back(20.54); // add 20.54
10.     prices.push_back(32.43); // add 32.43
11.     for(i = prices.begin(); i != prices.end(); i++)
12.       std::cout << *i << "  ";
13.     std::cout << std::endl;
14. }
```

12. Code 4.0, Line 12 will print
    a. Nothing
    b. 10.43   20.54   32.43
    c. 32.43  20.54  10.43
    d. 20.54  32.43  10.43
    e. None of the above

**Code5.0**

```cpp
#include <iostream>
#include <vector>

int main ()
{
    // constructors used in the same order as described above:
    std::vector<int> first;
    std::vector<int> second (4,100);
    std::vector<int> third (second.begin(),second.end());
    std::vector<int> fourth (std::move(third));
    std::vector<int> fifth (fourth);

    // the iterator constructor can also be used to construct from arrays:
    int myints[] = {16,2,77,29};
    std::vector<int> sixth (myints, myints + sizeof(myints) / sizeof(int) );

    std::cout << std::endl << "The contents of first are:";
    for (std::vector<int>::iterator it = first.begin(); it != first.end(); ++it)
           std::cout << ' ' << *it;

    std::cout << std::endl << "The contents of second are:";
    for (std::vector<int>::iterator it = second.begin(); it != second.end(); ++it)
           std::cout << ' ' << *it;

    std::cout << std::endl << "The contents of third are:";
    for (std::vector<int>::iterator it = third.begin(); it != third.end(); ++it)
           std::cout << ' ' << *it;

    std::cout << std::endl << "The contents of fourth are:";
    for (std::vector<int>::iterator it = fourth.begin(); it != fourth.end(); ++it)
           std::cout << ' ' << *it;

    std::cout << std::endl << "The contents of fifth are:";
    for (std::vector<int>::iterator it = fifth.begin(); it != fifth.end(); ++it)
           std::cout << ' ' << *it;

    std::cout << std::endl << "The contents of sixth are:";
    for (std::vector<int>::iterator it = sixth.begin(); it != sixth.end(); ++it)
           std::cout << ' ' << *it;

    std::cout << '\n';

    return 0;
}
```

13. Code 5.0, Line 16 prints
    a. 100 100 100 100
    b. 16 2 77 29
    c. 4 4 4 4
    d. All of the above
    e. None of the above
14. Code 5.0, Line 19 prints
    a. 100 100 100 100
    b. 16 2 77 29
    c. 4 4 4 4
    d. All of the above
    e. None of the above
15. Code 5.0, Line 22 prints
    a. 100 100 100 100
    b. 16 2 77 29
    c. 4 4 4 4
    d. All of the above
    e. None of the above
16. Code 5.0, Line 25 prints
    a. 100 100 100 100
    b. 16 2 77 29
    c. 4 4 4 4
    d. All of the above
    e. None of the above
17. Code 5.0, Line 28 prints
    a. 100 100 100 100
    b. 16 2 77 29
    c. 4 4 4 4
    d. All of the above
    e. None of the above
18. Code 5.0, Line 31 prints
    a. 100 100 100 100
    b. 16 2 77 29
    c. 4 4 4 4
    d. All of the above
    e. None of the above

**Code 6.0**

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <algorithm>
const int k = 10;
int main(int argc, char* argv[]) {
   1. std::vector<int> vec(5, 13);
   2. auto it = vec.end();
   3. auto initial = 0;
   4. for (std::vector<int>::iterator it2 = vec.begin(); it2 != vec.end(); ++it2)
   5. {
                  i.  *it2 = *it2 + initial;
                 ii.  initial = *it2;
   6. }
   7. std::cout << vec[2] << std::endl;
   8. std::cout << it - vec.begin() << std::endl;
   9. it--;
   10.std::cout << *it << std::endl;
   11.std::cout << vec.end() - vec.begin() << std::endl;
}
```

19. Code 6.0, The output of line 7 is:
    a. 5
    b. 13
    c. 39
    d. 65
    e. 26
20. The output of line 8 is:
    a. 5
    b. 13
    c. 2
    d. All of the above
    e. None of the above
21. The output of line 10 is:
    a. 5
    b. 13
    c. 39
    d. 65
    e. 26
22. The output of line 11 is:
    a. 5
    b. 13
    c. 39
    d. 65
    e. 26

**Code 7.0**

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <algorithm>
const int k = 10;
int main(int argc, char* argv[]) {
   1. std::vector<int> vec(5, 13);
   2. auto it = vec.begin() + 2;
   3. auto initial = 0;
   4. for (auto& it2 : vec )
   5. {
          a. it2 = it2 + initial;
          b. initial = it2;
   6. }
   7. std::cout << *it << std::endl;
   8. std::cout << it - vec.begin() << std::endl;
   9. it--;
   10. std::cout << *it << std::endl;
   11. std::cout << vec.end() - it << std::endl;

   12. for (auto it2 : vec )
   13. {
          a. it2 = it2 + initial;
          b. initial = it2;
   14. }

   15. std::cout << *it << std::endl;
}
```

23. Code 7.0, The output of line 7 is:
    a. 5
    b. 39
    c. 4
    d. 2
    e. 26
24. The output of line 8 is:
    a. 5
    b. 39
    c. 4
    d. 2
    e. 26

25. The output of line 10 is:
    a. 5
    b. 39
    c. 4
    d. 2
    e. 26
26. The output of line 11 is:
    a. 5
    b. 39
    c. 4
    d. 2
    e. 26

27. The output of line 15 is:
    a. 5
    b. 39
    c. 4
    d. 2
    e. 26