

Graded Quiz • 30 min

## Final Exam

Total points 42

1.	What is the C++11 meaning of the term &&?	1 point
	O pointer to 0	
	O No exceptions thrown	
	Move semantics	
	O Disable automatic conversion	
	O Infer type	
2.	What is the C++11 meaning of the term <b>nullptr</b> ?	1 point
	pointer to 0	
	O no exceptions thrown	
	O move semantics	
	O disable automatic conversion	
	O infer types	
3.	What is the C++11 meaning of the term auto	1 point
	O pointer to 0	
	O no exceptions thrown	
	O move semantics	
	O disable automatic conversion	
	infertype	
4.	What is the C++11 meaning of the term <b>explicit</b> ?	1 point
	O pointer to 0	
	O no exceptions thrown	
	O move semantics	
	disable automatic conversion	
	O infer type	
5.	What is the C++11 meaning of the term <b>noexcept</b> ?	1 point
	O pointer to 0	
	no exceptions thrown	
	O move semantics	
	O disable automatic conversion	
	O disable automatic conversion O infer type	
6.	O infer type	1 point
6.	O infer type  The header <future> in C++11 is used for:</future>	1 point
6.	O infer type  The header <future> in C++11 is used for: O type traits</future>	1 point
6.	O infer type  The header <future> in C++11 is used for: O type traits O random number generation</future>	1 point
6.	O infer type  The header <future> in C++11 is used for: O type traits</future>	1 point

	O none of these	
_	L. Euler invented:	
ι.		1 point
	O Galois theory	
	O Calculus	
	O quantum theory	
	graph theory	
	O none of these	
8.	D. Knuth analyzed or invented:	1 point
	<ul><li>alpha-beta</li></ul>	
	O Facebook likes	
	O Prolog	
	O C++ templates	
	O none of these	
9.	A new keyword in C++11 is:	1 point
	O thread	
	○ sizeof	
	• nullptr	
	O main	
	O none of these	
10	. A <b>try block</b> in C++:	1 point
	is only in Microsoft Windows code	
	is a scope that is followed by catch expressions	
	is used when there are virtual destructors	
	O initiates a new thread	
	O none of these	
11	. C++11 STL:	1 point
	O has no associative containers	
	O does not use templates	
	has a hash based map	
	O has a integration algorithm	
	O none of these	
12	HEX as a graph has its internal nodes:	1 point
	O with degree 2	
	O with degree 3	
	O with degree 4	
	O with degree 5	
	with degree 6	

	True	
	<ul><li>•</li></ul>	
	False	
	Two and also 2 Overlanded an author are always defined using static functions	
14.	True or false? Overloaded operators are always defined using static functions.	1 point
	O True	
	● False	
15.	True or false? All exceptions in C++ have as their base type the standard class <b>type std::exception</b> .	1 point
	O True	
	False	
	Town Class Wild Line and Lawrence Court	
16.	True or false? <b>WriteIn</b> is a new keyword in C++11.	1 point
	O True	
	False	
17.	True or false? In the expression $f(1) \parallel (!g(2))$ , and without knowing the return types of $f(1)$ and	1 point
	g(2), you can still assert that f(1) will always be evaluated before g(2)	
	(which may not get evaluated at all).	
	True	
	○ False	
18.	Using alpha-beta, can any LEAF nodes <i>not</i> be evaluated in the above tree?	1 point
	Yes	
	○ No	
19	When using =0 as the body of a function you are:	1 naint
		1 point
	Creating a null function	
	An abstract base class	
	O a syntax error	
	O an exception	
	O a zero return	
20.	The catch signature means:	1 point
	O Neveruse	
	O a syntax error	
	O either of n arguments	
	match any type	
	O catch the null exception	
21	In the following code segment, the type of <b>foobar</b> is:	1 1
<b>41</b> .	In the following code segment, the type of <b>foobar</b> is:	1 point
	<pre>1 list<int> data = {0,2,5,7,9}; 2 auto foobar = data.begin();</int></pre>	
	3 for(; foobar! = data.end();) 4   if (*foobar % 2 == 1)	
	5   foobar = data.erase(foobar); 6   else	
	7   ++foobar;	

int
unknown
a nullptr
Iist<int>::iterator

22. In the following code, the list will end up:

O none of these

1 point

```
1 list<int> data = {0,2,5,7,9};
2 auto foobar = data.begin();
3 for(; foobar! = data.end();)
4 if (*foobar % 2 == 1)
5 foobar = data.erase(foobar);
6 else
7 ++foobar;
```

O empty

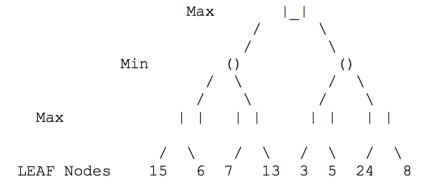
O having 5 elements

O having 3 elements

having 2 elements

O having 8 elements

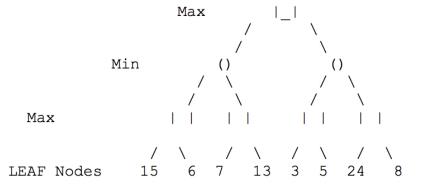
23. 1 point



For the above tree, what are the four Max values on the ply above the leaf nodes (from left to right). (Enter the 4 Max values as they appear, left to right, with a space separating them. For example, if your answers are 1, 2, 3 and 4, you would enter: 1 2 3 4.)

15 13 5 24

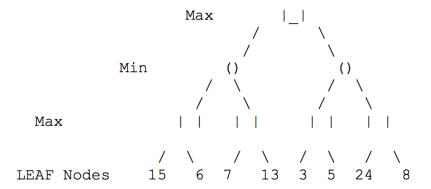
24.



For the above tree, what are the two Min values on the second ply (from left to right)? (Enter the 2 Min values as they appear, left to right, with a space separating them. For example, if your answers are 1 and 2, you would enter: 12.)

```
6738
```

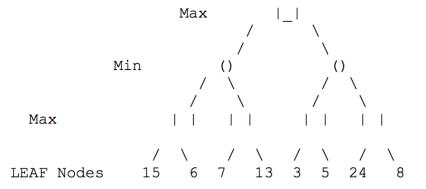
25.



What is the Max value for the root of the tree?

24

26.



In the above tree, what is one of the two leaf node values that need not have been evaluated, using alpha-beta? (Enter one of the two possible values.)

24

 $\textbf{27.} \ \ \textbf{What does the following print for the first *q?}$ 

```
#include <iostream>
     #include <vector>
     #include <algorithm>
     using namespace std;
     int main()
6
         int data[5]={1,7,46,9,6};
8
         vector<int> data_vec(data, data+5);
         int modulus = 3;
10
         //use of lambda's for predicates in find_if
11
         auto q = find_if( data_vec.begin(),data_vec.end(),
12
         [](int elem)->bool{ if (elem % 2 ==0 )return true;
                                                                                      else return false;}
13
14
15
        cout << "first *q " << *q << endl;
16
17
         //next lambda has a capture by value
18
         q = find_if( data_vec.begin(),data_vec.end(),
20
     [=](int elem)->bool{ if (elem % modulus ==0 )
21
                        return true;
```

46

28. What does the following print for the second \*q?

1 point

```
#include <iostream>
     #include <vector>
     #include <algorithm>
4
     using namespace std;
    int main()
6
         int data[5]={1,7,46,9,6};
8
         vector<int> data_vec(data, data+5);
9
         int modulus = 3;
        //use of lambda's for predicates in find_if
10
11
         auto q = find_if( data_vec.begin(),data_vec.end(),
12
         [](int elem)->bool{ if (elem % 2 ==0 )return true;
                                                                                       else return false;}
13
14
15
         cout << "first *q " << *q << endl;</pre>
16
17
         //next lambda has a capture by value
18
19
         q = find_if( data_vec.begin(),data_vec.end(),
20
     [=](int elem)->bool{ if (elem % modulus ==0 )
21
                             return true;
22
                        else return false;}
23
24
25
         cout << "second *q " << *q << endl;
26
27
```

9

29. What does the first line of the following print?

1 point

```
#include <iostream>
     #include <vector>
     #include <algorithm>
    using namespace std;
     int main()
6
         vector<int> data(5,1);
8
9
         int sum {0};
10
11
         cout << sum << endl:
12
         for(auto element : data)
13
             sum += element;
14
         cout << sum << endl;
15
16
17
         for(auto p = ++data.begin(); p != --data.end(); ++p)
18
             sum += *p;
         cout << sum << endl;</pre>
19
20
         sum = 0;
21
         data.push_back(2);
22
23
         data.push_back(3);
24
         for(auto element : data)
25
26
              sum += element;
27
         cout << sum << endl;
28
         \verb|cout << accumulate(data.begin(), data.end(), data[0]) << endl;\\
29
30
31
```

0

**30** A key nurnose of move semantics is:

They purpose of more semanticals.

- Type correctness
- O clearer code
- efficiency
- O functional semantic
- O none of these

31. What does the second line of the following print?

1 point

```
#include <iostream>
     #include <vector>
     #include <algorithm>
 4 using namespace std;
     int main()
          vector<int> data(5,1);
8
         int sum {0};
9
10
         cout << sum << endl:
11
12
13
         for(auto element : data)
14
              sum += element;
         cout << sum << endl;
15
16
17
         for(auto p = ++data.begin(); p != --data.end(); ++p)
              sum += *p;
18
         cout << sum << endl;
19
20
21
         sum = 0;
         data.push_back(2);
22
23
         data.push_back(3);
24
25
         for(auto element : data)
26
              sum += element;
         cout << sum << endl;</pre>
27
28
29
          \verb|cout << accumulate(data.begin(), data.end(), data[0]) << \verb|endl|;|\\
30
31
```

```
5
```

32. What does the third line of the following print?

```
#include <iostream>
     #include <vector>
     #include <algorithm>
    using namespace std;
     int main()
         vector<int> data(5,1);
         int sum {0};
10
11
         cout << sum << endl;</pre>
12
13
        for(auto element : data)
14
             sum += element;
15
        cout << sum << endl;
16
17
         for(auto p = ++data.begin(); p != --data.end(); ++p)
             sum += *p;
18
         cout << sum << endl;
19
20
         sum = 0;
21
22
         data.push_back(2);
         data.push_back(3);
23
24
         for(auto element : data)
25
26
             sum += element;
27
         cout << sum << endl;</pre>
28
         cout << accumulate(data.begin(), data.end(), data[0]) << endl;
29
30
31
```

```
0
```

33. What does the fourth line of the following print?

```
1 point
```

```
int sum {0};
10
11
        cout << sum << endl;</pre>
12
13
        for(auto element : data)
            sum += element:
14
        cout << sum << endl;
15
16
        17
18
19
        cout << sum << endl;</pre>
20
        sum = 0;
21
        data.push_back(2);
22
23
        data.push_back(3);
24
25
        for(auto element : data)
26
            sum += element;
27
        cout << sum << endl;</pre>
28
29
        \verb|cout| << | accumulate(data.begin(), data.end(), data[0]) << | end|; \\
30
31
```

10

34. What does the fifth line of the following print?

```
1 point
```

```
#include <iostream>
     #include <vector>
     #include <algorithm>
4
     using namespace std;
     int main()
8
         vector<int> data(5,1);
9
         int sum \{0\};
10
11
         cout << sum << endl;</pre>
12
13
         for(auto element : data)
14
             sum += element;
15
         cout << sum << endl;</pre>
16
17
         for(auto p = ++data.begin(); p != --data.end(); ++p)
18
             sum += *p;
19
         cout << sum << endl;</pre>
20
21
22
         data.push_back(2);
23
         data.push_back(3);
24
         for(auto element : data)
25
26
             sum += element;
27
         cout << sum << endl;
28
29
         cout << accumulate(data.begin(), data.end(), data[0]) << endl;</pre>
30
31
```

```
11
```

35. What does the first line of the following print?

```
1 #include <iostream>
2 using namespace std;
3
4 class Animal {
5 public:
```

```
virtual void speak()=0;
        virtual void purr() { cout << "Purr\n"; }</pre>
8
9
    class Cat : public Animal {
10
     public:
11
       void speak() { cout << "Meow\n";purr(); }</pre>
12
13
   class Lion : public Cat {
14
     public:
15
        void speak() { cout << "ROAR\n"; }</pre>
16
         void purr() { cout << "ROAR\n"; }</pre>
17
18
     int main() {
19
      Animal* c = new Cat();
       Cat napster;
21
      Lion googly;
23
      c->speak();
25
      napster.speak();
      googly.speak();
26
27
      return 0;
28
29
```

Meou

**36.** What does the second line of the following print?

1 point

```
#include <iostream>
2 using namespace std;
4 class Animal {
5
     public:
        virtual void speak()=0;
6
         virtual void purr() { cout << "Purr\n"; }</pre>
8   };
9   class Cat : public Animal {
10 public:
       void speak() { cout << "Meow\n";purr(); }</pre>
11
12 };
13
    class Lion : public Cat {
14
     public:
        void speak() { cout << "ROAR\n"; }
void purr() { cout << "ROAR\n"; }</pre>
15
16
17
18
    int main() {
      Animal* c = new Cat();
19
20
       Cat napster;
21
       Lion googly;
22
23
       c->speak();
24
25
26
       googly.speak();
27
28
29
```

Purr

37. What does the third line of the following print?

```
#include <iostream>
 2 using namespace std;
 4 class Animal {
    public:
        virtual void speak()=0;
        virtual void purr() { cout << "Purr\n"; }</pre>
8 };
    class Cat : public Animal {
10 public:
      void speak() { cout << "Meow\n";purr(); }</pre>
11
12 };
13
    class Lion : public Cat {
14
     public:
        void speak() { cout << "ROAR\n"; }</pre>
15
        void purr() { cout << "ROAR\n"; }</pre>
16
17
    };
    int main() {
18
     Animal* c = new Cat();
19
```

```
20
      Cat napster;
21
      Lion googly;
22
23
      c->speak();
24
25
      napster.speak();
26
      googly.speak();
27
      return 0;
28
29
```

Meow

38. What does the fourth line of the following print?

1 point

```
#include <iostream>
     using namespace std;
 4 class Animal {
     public:
        virtual void speak()=0;
          virtual void purr() { cout << "Purr\n"; }</pre>
     class Cat : public Animal {
10 public:
       void speak() { cout << "Meow\n";purr(); }</pre>
11
12 };
13
     class Lion : public Cat {
     public:
14
         void speak() { cout << "ROAR\n"; }
void purr() { cout << "ROAR\n"; }</pre>
15
16
17
     };
     int main() {
   Animal* c = new Cat();
18
19
20
       Cat napster;
       Lion googly;
21
22
23
       c->speak();
24
25
       napster.speak();
26
       googly.speak();
27
       return 0;
28
29
```

Purr

39. What does the fifth line of the following print?

```
#include <iostream>
     using namespace std;
    class Animal {
    public:
        virtual void speak()=0;
        virtual void purr() { cout << "Purr\n"; }</pre>
9
    class Cat : public Animal {
10
    public:
11
      void speak() { cout << "Meow\n";purr(); }</pre>
12
13
    class Lion : public Cat {
14
     public:
15
        void speak() { cout << "ROAR\n"; }</pre>
16
        void purr() { cout << "ROAR\n"; }</pre>
17
18
     int main() {
19
      Animal* c = new Cat();
       Cat napster;
20
21
      Lion googly;
22
23
      c->speak();
24
25
       napster.speak();
26
       googly.speak();
27
       return 0;
28
29
```

40. The safest cast in C++ is considered:	1 point
O (type)	
static_assert	
○ (void*)	
○ static_cast	
O reinterpret_cast	
<b>41.</b> The MST for an undirected connected graph of N nodes where all weights are cost C has:	1 point
O a value that cannot be determined	
○ a value of 2*N*C	
a value of 2*N*(C-1)	
O a value of (N-1)*C	
<b>42.</b> True or false? Overloaded operators are always defined using static functions.	1 point
○ True	
False	
Coursera Honor Code Learn more	
I, Amanjeet Kumar, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.	
Submit Save draft	
🖰 Like 🖓 Dislike 📙 Report an issue	