1. What is the C++11 meaning of the term **lvl**? 1point

- pointer to 0
- No exceptions thrown
- Move semantics
- Disable automatic conversion
- Infer type

2. What is the C++11 meaning of the term **nullptr**? 1point

- pointer to 0
- no exceptions thrown
- move semantics
- disable automatic conversion
- infer type

3. What is the C++11 meaning of the term **auto**: 1point

- pointer to 0
- no exceptions thrown
- move semantics
- disable automatic conversion
- infer type

4. What is the C++11 meaning of the term **explicit**? 1point

- pointer to 0
- no exceptions thrown
- move semantics
- disable automatic conversion
- infer type

5. What is the C++11 meaning of the term **noexcept**? 1point

- pointer to 0
- no exceptions thrown
- move semantics
- disable automatic conversion
- infer type

6. The header **<future>** in C++11 is used for: 1point

- type traits
- random number generation
- concurrency
- stack loading
- none of these

7. L-Ester invented: 1point

- Calculus theory
- Calculus
- quantum theory
- graph theory
- none of these

8. D-Knuth analyzed or invented: 1point

- alpha-beta
- Facebook likes
- Pricing
- C++ templates
- none of these

9. A new keyword in C++11 is: 1point

- thread
- sizeof
- nullptr
- main
- none of these

10. A **try block** in C++: 1point

- is only in Microsoft Windows code
- is a scope that is followed by catch expressions
- is used when there are virtual destructors
- initializes a new thread
- none of these

11. C++11 STL: 1point

- has no associative containers
- does not use templates
- has a hash based map
- has a integration algorithm
- none of these

12. HCS as a graph has its internal nodes: 1point

- with degree 2
- with degree 3
- with degree 4
- with degree 5
- with degree 6

13. True or false? When you retrieve an exception, its type and value is convertible to **int**. 1point

- True
- False

14. True or false? Overloaded operators are always defined using static functions. 1point

- True
- False

15. Prior to false? All exceptions in C++ have as their base type the standard class **type std::exception**. 1point

- True
- False

16. True or false? **Writeln** is a new keyword in C++11. 1point

- True
- False

17. True or false? In the expression **f(i) || g(i)**, and without knowing the return types of **f(i)** and **g(i)**, you can still assert that **f(i)** will always be evaluated before **g(i)**. (which may not get evaluated at all). 1point

- True
- False

18. Using alpha-beta, can any LEAF nodes not be evaluated in the above tree? 1point

- Yes.
- No.

19. When using **=0** in the body of a function you are: 1point

- Creating a null function
- An abstract base class
- a syntax error
- an exception
- a zero return

20. The catch signature **...** means: 1point

- Never use
- a syntax error
- either of or arguments
- match any type
- catch the null exception

21. In the following code segment, the type of **fbarber** is: 1point

```
int(int data = 2)(int,int);
auto fbarber = data.begin();
if (fbarber == data.end() )
if (fbarber $ 1 == 1)
    fbarber = data.erase(fbarber);
else
    ++fbarber;
```

- int
- unknown
- a nullptr
- dynamic/ptr/ptr
- none of these

22. In the following code, the list will end up: 1point

```
int(int data = 2)(int,int);
auto fbarber = data.begin();
if (fbarber == data.end() )
if (fbarber $ 1 == 1)
    fbarber = data.erase(fbarber);
else
    ++fbarber;
```

- empty
- having 3 elements
- having 3 elements
- having 3 elements
- having 4 elements

23.

For the above tree, what are the first **Max** values on the ply above the leaf nodes (from left to right). (Enter the **Max** values as they appear, left to right, with a space separating them. For example, if your answers are **5** & **2** and **4**, you would enter: **5 2 4**).

15 13 3 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 **5** and **2**, you would enter: **5 2**).

15 5

25.

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

13

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

27. What does the following print for the first **%d**: 1point

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main()
{
    int data[]={1,5,4,6,3};
    vector<int> data_vec(data, data+5);
    int mybiter = 0;
    int(int,int)(int,int) for_predicate = [](int,int
auto a = find_if(data_vec.begin(),data_vec.end(),
    [](int x){ return( x % 2 == 1); });                    else {
    if(a != data_vec.end() && *a == 1)
        return true;
    else return false;
    } );
    cout << "First %d " << *a << endl;

    //next lambda has a capture by value

    a = find_if(data_vec.begin(),data_vec.end(),
[](int x){ return( x % 2 == 1); });                        else {
    if(a != data_vec.end() && *a == 1)
        return true;
    else return false;
    } );
    cout << "second %d " << *a << endl;
}
```

45

28. What does the following print for the second **%d**: 1point

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main()
{
    int data[]={1,5,4,6,3};
    vector<int> data_vec(data, data+5);
    int mybiter = 0;
    int(int,int)(int,int) for_predicate = [](int,if
auto a = find_if(data_vec.begin(),data_vec.end(),
    [](int x){ return( x % 2 == 1); });                    else {
    if(a != data_vec.end() && *a == 1)
        return true;
    else return false;
    } );
    cout << "First %d " << *a << endl;

    //next lambda has a capture by value

a = find_if(data_vec.begin(),data_vec.end(),
[](int x){ return( x % 2 == 1); });                        else {
    if(a != data_vec.end() && *a == 1)
        return true;
    else return false;
    } );
    cout << "second %d " << *a << endl;
}
```

5

29. What does the first line of the following print? 1point

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

int main()
{
    vector<int> data(5,1);
    int sum (0);

    cout << sum << endl;

    for(int element : data)
        sum += element;
    cout << sum << endl;

    for(int p = rotate(begin(), p != --data.end(), rep)
        sum += *p;
    cout << sum << endl;

    sum = 0;
    data.push_back(1);
    data.push_back(1);

    for(int element : data)
        sum += element;
    cout << sum << endl;

    sum = accumulate(data.begin(), data.end(), data[0]) << endl;
}
```

0

30. A key purpose of move semantics is: 1point

- Type correctness
- cleaner code
- efficiency
- functional semantic
- none of these

31. What does the second line of the following print? 1point

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

int main()
{
    vector<int> data(5,1);
    int sum (0);

    cout << sum << endl;

    for(int element : data)
        sum += element;
    cout << sum << endl;

    for(int p = rotate(begin(), p != --data.end(), rep)
        sum += *p;
    cout << sum << endl;

    sum = 0;
    data.push_back(1);
    data.push_back(1);
```

```
        sum += element;
    cout << sum << endl;

    cout << accumulate(data.begin(), data.end(), data[0]) << endl;
}
```

5

32. What does the third line of the following print? **1 point**

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

int main()
{
    vector<int> data(5,1);
    int sum (0);

    cout << sum << endl;

    for(auto element : data)
        sum += element;
    cout << sum << endl;

    for(auto p = --data.begin(); p != --data.end(); ++p)
        sum += *p;
    cout << sum << endl;

    sum = 0;
    data.push_back(1);
    data.push_back(1);

    for(auto element : data)
        sum += element;
    cout << sum << endl;

    cout << accumulate(data.begin(), data.end(), data[0]) << endl;
}
```

8

33. What does the fourth line of the following print? **1 point**

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

int main()
{
    vector<int> data(5,1);
    int sum (0);

    cout << sum << endl;

    for(auto element : data)
        sum += element;
    cout << sum << endl;

    for(auto p = --data.begin(); p != --data.end(); ++p)
        sum += *p;
    cout << sum << endl;

    sum = 0;
    data.push_back(1);
    data.push_back(1);

    for(auto element : data)
        sum += element;
    cout << sum << endl;

    cout << accumulate(data.begin(), data.end(), data[0]) << endl;
}
```

10

34. What does the fifth line of the following print? **1 point**

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

int main()
{
    vector<int> data(5,1);
    int sum (0);

    cout << sum << endl;

    for(auto element : data)
        sum += element;
    cout << sum << endl;

    for(auto p = --data.begin(); p != --data.end(); ++p)
        sum += *p;
    cout << sum << endl;

    sum = 0;
    data.push_back(1);
    data.push_back(1);

    for(auto element : data)
        sum += element;
    cout << sum << endl;

    cout << accumulate(data.begin(), data.end(), data[0]) << endl;
}
```

11

35. What does the first line of the following print? **1 point**

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

class Animal {
public:
    virtual void speak();=0;
    virtual  void purr() { cout << "Purrin"; }
};
class Cat : public Animal {
public:
    void speak() { cout << "Meow\n";purr(); }
};
class Lion : public Cat {
public:
    void speak() { cout << "ROAR\n"; }
    void purr() { cout << "ROAR\n"; }
};
int main() {
    Animal* c = new Cat();
    Cat napster;
    Lion googly;

    c->speak();

    napster.speak();
    googly.speak();
    return 0;
}
```

Meow

36. What does the second line of the following print? **1 point**

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

class Animal {
public:
    virtual void speak();=0;
    virtual  void purr() { cout << "Purrin"; }
};
class Cat : public Animal {
public:
    void speak() { cout << "Meow\n";purr(); }
};
class Lion : public Cat {
public:
    void speak() { cout << "ROAR\n"; }
    void purr() { cout << "ROAR\n"; }
};
int main() {
    Animal* c = new Cat();
    Cat napster;
    Lion googly;

    c->speak();

    napster.speak();
    googly.speak();
    return 0;
}
```

Purr

37. What does the third line of the following print? **1 point**

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

class Animal {
public:
    virtual void speak();=0;
    virtual  void purr() { cout << "Purrin"; }
};
class Cat : public Animal {
public:
    void speak() { cout << "Meow\n";purr(); }
};
class Lion : public Cat {
public:
    void speak() { cout << "ROAR\n"; }
    void purr() { cout << "ROAR\n"; }
};
int main() {
    Animal* c = new Cat();
    Cat napster;
    Lion googly;

    c->speak();

    napster.speak();
    googly.speak();
    return 0;
}
```

Meow

38. What does the fourth line of the following print? **1 point**

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

class Animal {
public:
    virtual void speak();=0;
    virtual  void purr() { cout << "Purrin"; }
};
class Cat : public Animal {
public:
    void speak() { cout << "Meow\n";purr(); }
};
class Lion : public Cat {
public:
    void speak() { cout << "ROAR\n"; }
    void purr() { cout << "ROAR\n"; }
};
int main() {
    Animal* c = new Cat();
    Cat napster;
    Lion googly;

    c->speak();

    napster.speak();
    googly.speak();
    return 0;
}
```

Purr

39. What does the fifth line of the following print? **1 point**

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

class Animal {
public:
    virtual void speak();=0;
    virtual  void purr() { cout << "Purrin"; }
};
class Cat : public Animal {
public:
    void speak() { cout << "Meow\n";purr(); }
};
class Lion : public Cat {
public:
    void speak() { cout << "ROAR\n"; }
    void purr() { cout << "ROAR\n"; }
};
int main() {
    Animal* c = new Cat();
    Cat napster;
    Lion googly;

    c->speak();

    napster.speak();
    googly.speak();
    return 0;
}
```

ROAR

40. The safest cast in C++ is considered: **1 point**

- (void*)
- static_cast
- reinterpret_cast
- static_assert
- (type)

41. The MST for an undirected connected graph of N nodes where all weights are cost C has: **1 point**

- a value of (N-1)*C
- a value of 2*N*(C-1)
- a value that cannot be determined
- a value of 2*N*C
- a value of N*C

42. True or false? Overloaded operators are always defined using static functions. **1 point**

- true
- False

Coursera Honor Code Learn more

☐ I, **WINGZHENG HUANG**, 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