← **Back** **Final Exam**
Graded Quiz • 30 min

Due May 30, 12:29 PM IST

Final Exam

Total points 42

1. What is the C++11 meaning of the term &&?

1 point

○ pointer to 0

○ No exceptions thrown

● Move semantics

○ Disable automatic conversion

○ Infer type

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

1 point

● pointer to 0

○ no exceptions thrown

○ move semantics

○ disable automatic conversion

○ infer types

3. What is the C++11 meaning of the term auto

1 point

○ pointer to 0

○ no exceptions thrown

○ move semantics

○ disable automatic conversion

● infer type

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

1 point

○ pointer to 0

○ no exceptions thrown

○ move semantics

● disable automatic conversion

○ infer type

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

1 point

○ pointer to 0

● no exceptions thrown

○ move semantics

○ disable automatic conversion

○ infer type

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

1 point

○ type traits

○ random number generation

● concurrency

○ stock trading

○ none of these

7. L. Euler invented: **1 point**

○ Galois theory

○ Calculus

○ quantum theory

● graph theory

○ none of these

8. D. Knuth analyzed or invented: **1 point**

● alpha-beta

○ Facebook likes

○ Prolog

○ C++ templates

○ none of these

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

○ thread

○ sizeof

● nullptr

○ main

○ none of these

10. A **try block** 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

○ initiates a new thread

○ none of these

11. C++11 STL: **1 point**

○ has no associative containers

○ does not use templates

● has a hash based map

○ has a integration algorithm

○ none of these

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

○ with degree 2

○ with degree 3

○ with degree 4

○ with degree 5

● with degree 6

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

○ True

● False

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

1 point

○ True

● False

15. True or false? All exceptions in C++ have as their base type the standard class **type std::exception**.

1 point

○ True

● False

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

1 point

○ True

● False

17. True or false? In the expression **f(1)** || (**!g(2)**), and without knowing the return types of **f(1)** and **g(2)**, you can still assert that **f(1)** will always be evaluated before **g(2)** (which may not get evaluated at all).

1 point

● True

○ False

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

1 point

● Yes

○ No

19. When using **=0** as the body of a function you are:

1 point

○ Creating a null function

● An abstract base class

○ a syntax error

○ an exception

○ a zero return

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

1 point

○ Never use

○ a syntax error

○ either of n arguments

● match any type

○ catch the null exception

21. In the following code segment, the type of **foobar** is:

1 point

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

○ int

○ unknown

○ a nullptr

● list<int>::iterator

○ none of these

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

1 point

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

○ empty

○ having 5 elements

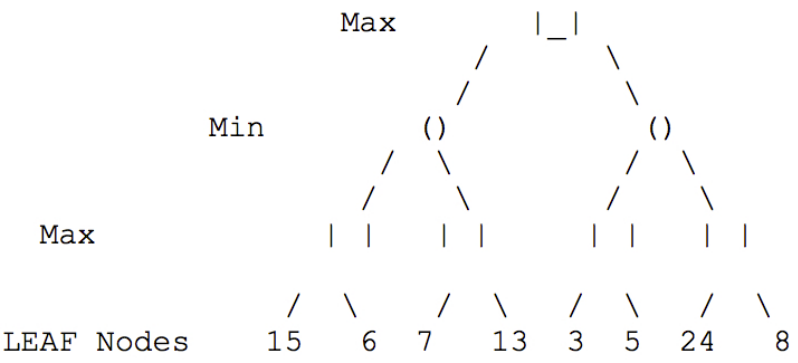
○ having 3 elements

● having 2 elements

○ having 8 elements

23.

1 point

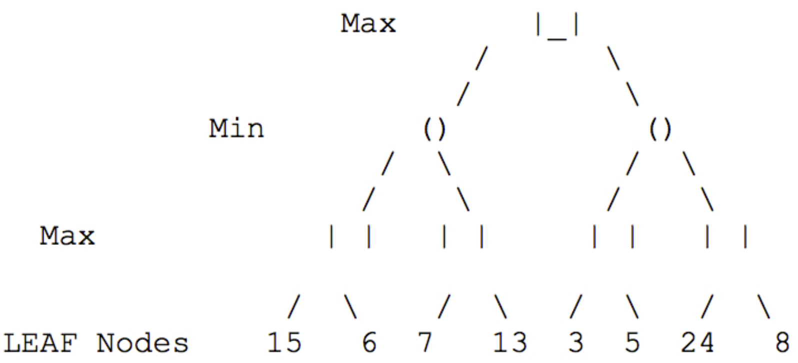


For the above tree, what are the four **Max** values on the ply above the leaf nodes (from left to right). (Enter the 4 Max values as they appear, left to right, with a space separating them. For example, if your answers are **1**, **2**, **3** and **4**, you would enter: **1 2 3 4**.)

15 13 5 24

24.

1 point

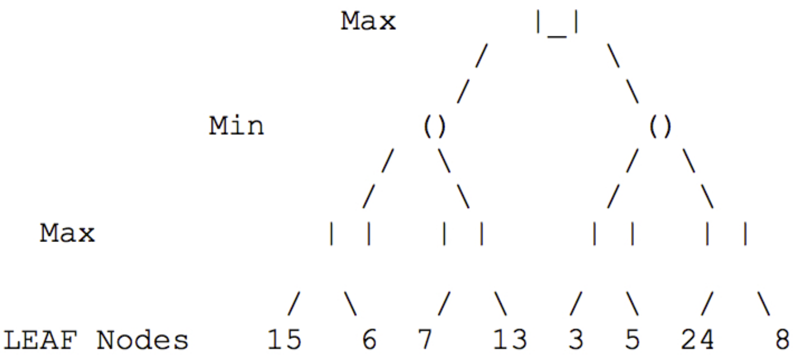


For the above tree, what are the two **Min** values on the second ply (from left to right)? (Enter the 2 Min values as they appear, left to right, with a space separating them. For example, if your answers are **1** and **2**, you would enter: **1 2**.)

6 7 3 8

25.

1 point

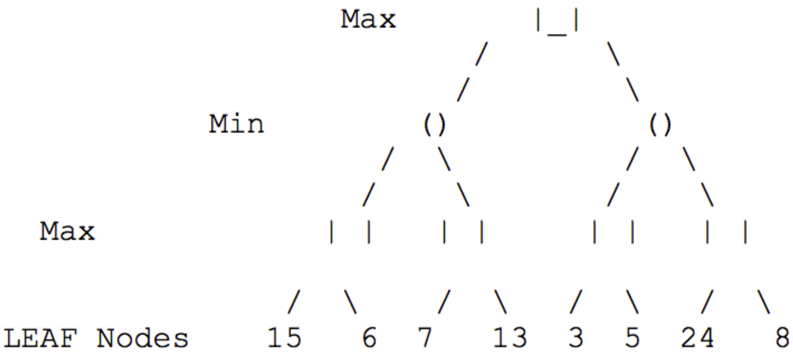


What is the **Max** value for the root of the tree?

24

26.

1 point



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 ***q**?

1 point

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main()
{
    int data[5]={1,7,46,9,6};
    vector<int> data_vec(data, data+5);
    int modulus = 3;
    //use of lambda's for predicates in find_if
    auto q =  find_if( data_vec.begin(),data_vec.end(),
     [](int elem)->bool{ if (elem % 2 ==0 )return true;                          else  return false;}
    ) ;

    cout << "first *q " << *q << endl;

    //next lambda has a capture by value

    q =  find_if( data_vec.begin(),data_vec.end(),
[=](int elem)->bool{ if (elem % modulus ==0 )
                        return true;
```

```
                     else  return false;}
    ) ;

    cout << "second *q " << *q << endl;
}
```

46

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

1 point

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main()
{
    int data[5]={1,7,46,9,6};
    vector<int> data_vec(data, data+5);
    int modulus = 3;
    //use of lambda's for predicates in find_if
    auto q =  find_if( data_vec.begin(),data_vec.end(),
     [](int elem)->bool{ if (elem % 2 ==0 )return true;                            else  return false;}
    ) ;

    cout << "first *q " << *q << endl;

    //next lambda has a capture by value

    q =  find_if( data_vec.begin(),data_vec.end(),
[=](int elem)->bool{ if (elem % modulus ==0 )
                          return true;
                     else  return false;}
    ) ;

    cout << "second *q " << *q << endl;
}

```

9

29. What does the first 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(2);
    data.push_back(3);

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

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

```

0

30. A key purpose of move semantics is:

1 point

30. A key purpose of move semantics is:

1 point

○ Type correctness

○ clearer code

● efficiency

○ functional semantic

○ none of these

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

1 point

```cpp
#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(2);
    data.push_back(3);

    for(auto element : data)
        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?

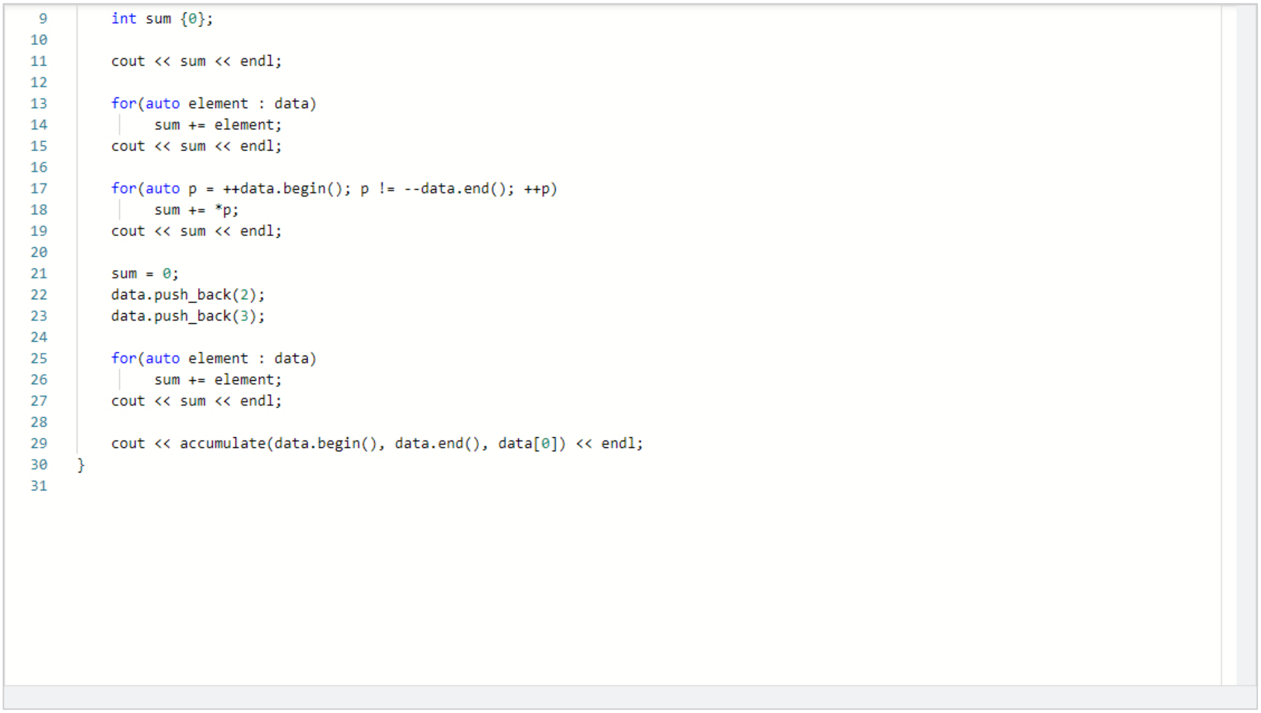
1 point

```cpp
#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(2);
    data.push_back(3);

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

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

```

0

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

1 point

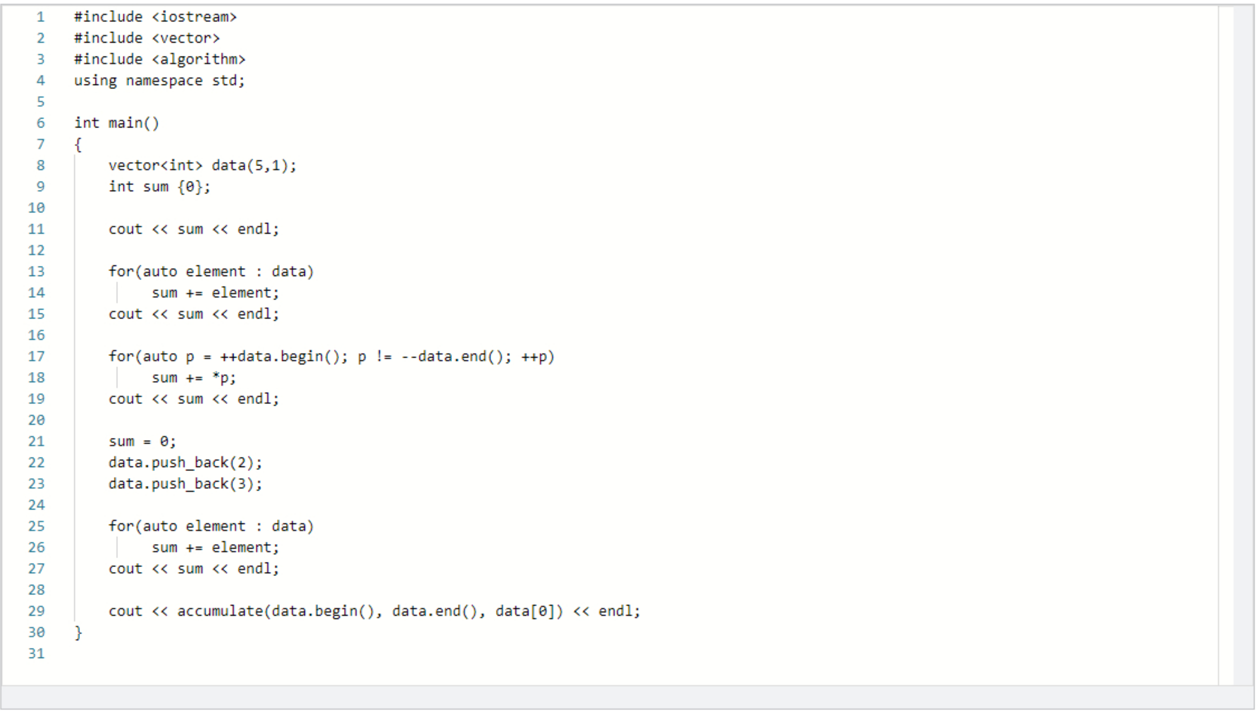


```
    int sum {0};

    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(2);
    data.push_back(3);

    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

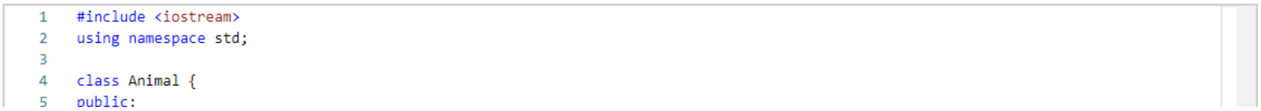


```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

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(2);
    data.push_back(3);

    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 << "Purr\n"; }
};
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 << "Purr\n"; }
};
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 << "Purr\n"; }
};
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 << "Purr\n"; }
};
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 << "Purr\n"; }
};
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

○ (type)

● static_assert

○ (void*)

○ static_cast

○ reinterpret_cast

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

1 point

○ a value that cannot be determined

○ a value of 2*N*C

● a value of N*C

○ a value of 2*N*(C-1)

○ a value of (N-1)*C

42. 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