ACCU 2019

C++ Pub Quiz

by Felix Petriconi with Sean Parent

Sponsored by Bloomberg

++

Sponsored by Bloomberg

A 90 minute quiz session at ACCU
Conservatory, Marriott Hotel, Bristol
16:00-17:30, 2019-04-11

Acknowledgement:

Olve Maudal developed the C++ Pub Quiz.

He encouraged me to pickup and continue this format.

The question for all code snippets is:
What will actually happen
when it would be compiled and executed?

All examples produce the same result compiled with
-O2 -std=c++17

gcc 8.2.0
clang 7.0.0
Visual Studio 2017 Update 9

None of the code examples contain *Undefined Behaviour!*

All the code snippets do indeed compile, link and run.
There are no missing semicolons or syntax errors in the snippets.
The output is always a straightforward sequence of non-whitespace characters.

Disclaimer: The code snippets here are all crap!

And when I say crap, I mean crap!

It shows examples of how to write code - or not to write code.
This is just for fun.
Remember, this is **not** about c++, nor g++, it is about:

++

accu 2019

++

C++ Pub Quiz

Sponsored by
Bloomberg

#	Answer	Notes	Score	Bonus
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Team Name:

Start Bonus	Score	Bonus	Total

accu
2019

🍺++

C++ Pub Quiz

Sponsored by
Bloomberg

#	Answer	Notes	Score	Bonus
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Team Name: Team Marvin

Start Bonus	Score	Bonus	Total
10			

accu 2019

10 points as start bonus
3 points for each correct answer
0 point for incorrect answer
-1 point for no answer
For many of the questions there are bonus points.

Questions?

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

template <typename T> void P(T const& x) { cout << x; }

int main() {
    int a[]{1,2,3,4};
    P(0);
    for (auto x : a)
        P(x);
    P(6);
}
```

++

C++ Pub Quiz

Sponsored by
Bloomberg

#	Answer	Notes	Score	Bonus
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Team Name: Team Marvin

Start Bonus	Score	Bonus	Total
10			

accu 2019

++

C++ Pub Quiz

Sponsored by Bloomberg

#	Answer	Notes	Score	Bonus
0	012346		3	
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Team Name: Team Marvin

Start Bonus	Score	Bonus	Total
10			

accu 2019

🍺++

C++ Pub Quiz

Sponsored by Bloomberg

#	Answer	Notes	Score	Bonus
0	012346	auto x : a, copy by value	3	1
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Team Name: Team Marvin

Start Bonus	Score	Bonus	Total
10			

accu 2019

```
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<pair<int, char>> v = { {1,'b'}, {2,'a'}, {1,'c'}, {2,'b'}};
    stable_sort(begin(v), end(v), [](auto const& x, auto const& y) {
        return y.first == min(x.first, y.first);
    });
    for (auto x : v) { cout << x.second; }
    stable_sort(begin(v), end(v), [](auto const& x, auto const& y) {
        return y.first == max(x.first, y.first);
    });
    for (auto x : v) { cout << x.second; }
}
```

#2

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

template <typename... T> auto P(T... x) { (cout << ... << x); return get<0>(make_tuple(x...)); }

template <typename T>
struct w {
    T _v;
    w(T v): _v(v) {}
    operator T() const { return _v; }
};

template <typename T, typename U>
int operator,(T a, U b) { return P(a + b); }

template <typename X, typename Y, typename Z>
Y f(X x, Y y, Z z) {
    P('x');P(x);P(y,z);P('y');P((y,z));
    return P(y),z;
}

int a{1}, c{2};
w b{3}, d{4};

int main() {
    P(f(a, b, P(c,d)));
}
```

#3

```
#include <iostream>
using namespace std;
template <typename T> void P(const T& x) { cout << x; }

struct test {
    int _v;
    test(int v = 0) : _v(v) {}
    auto run() & { return _v; }
    auto run() const& { return _v+1; }
    auto run() && { return _v+2; }

    auto operator|(int v) & { return test(_v+v); }
    auto operator|(int v) const& { return test(_v+v); }
    auto operator|(int v) && { return test(_v+v); }

    auto operator&(int v) & { return run() + v; }
    auto operator&(int v) const& { return run() + v; }
    auto operator&(int v) && {
        return std::move(*this).run() + v;
    }

    operator int() const { return _v; }
};
```

```
int main() {
    test a1;
    P(a1.run());
    const test a2;
    P(a2.run());
    P(test().run());
    P(a1 | 2);
    P(a2 | 3);
    P(a1 & 2);
    P(a2 & 3);
    P(test{4} | 1);
    P(test{5} | 2 | 3);
    P(a2 | 1 | 2 & 3);
}
```

#4

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

template <typename T> void P(T const& x) { cout << x; }

template <typename...T>
auto s1(T... t) { return (t + ... ); }

template <typename...T>
auto s2(T... t) { return (... *= t); }

template <typename...T>
auto s3(T... t) { return (P(t), ... , P(1)), 1; }

template <typename...T>
auto s4(T... t) { int v{1}; return (v &= ... &= t); }

int main() {
    P(s1(1,2,3));
    P(s2(4,5,6));
    P(s3(7,8,9));
    P(s4(1,2,4));
}
```

#5

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

void P(int x) { cout << x; }

template <size_t I>
struct parent
{
    size_t value = I;
};

template <size_t... I>
struct child : parent<I>...
{
    operator int() const { return ( this->parent<I>::value + ...); }

    template <size_t...J, typename... T>
    void set(index_sequence<J...>, T... t) {
        P(*initializer_list<int>{(this->parent<I-J>::value *= t, 0)...}.begin());
    }

    template <typename ...T>
    void set(T... t) {
        set(make_index_sequence<sizeof...(T)>(), t...);
    }
};

int main() {
    child<1,2,3> b;
    P(b);
    b.set(4,5,6);
    P(b);
    b.set(7,8,9);
    P(b);
}
```

#6

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

template <typename T> void P(T const& x) { cout << x; }

template<typename P>
void foo(string const& str, P& p) {
    for (auto ch : str)
        p(ch);
}

int main() {
    int num = 1;
    string str("abc");
    auto f = [=](char ch) mutable { P(num++); P(ch); };
    foo(str, f);
    foo(str, f);
    P(num);
}
```

#7

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

template<char S, size_t C>
struct R {
    string d;
    size_t c = 0;

    struct E {
        size_t& c;
        bool operator()(string::iterator it) const {
            if (*it == S) ++c;
            return c != C;
        }
    };

    auto begin() { return d.begin(); }
    auto end() { return E{c}; }

    friend bool operator!=(string::iterator it, E const& p) { return p(it); }
};

int main() {
    auto t = "1201230012345000123456123456 7";
    for (auto c : R<'0', 1>{t}) cout << c;
    for (auto c : R<'1', 2>{t}) cout << c;
    for (auto c : R<'2', 3>{t}) cout << c;
}
```

accu 2019

```
#include <algorithm>
#include <iostream>
#include <iterator>
#include <numeric>

using namespace std;

int main() {
    int a[] = {2, 3, 4, 5, 2, 1, 5};
    nth_element(begin(a), &a[4], end(a));
    cout << a[5] << reduce(begin(a), &a[5]);
}
```

#8

accu
2019

#9

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

template <typename T> void P(T const& v) { cout << v; }

template <typename T>
struct w
{
    T _v;
    w(T v) : _v(move(v)) { P('a'); }
    w(w const& x): _v(x._v) { P(_v); P('b'); }
    w(w&& x) { *this = move(x); P(_v); P('c'); }
    w& operator=(w const& x) {
        auto tmp = x; *this = move(tmp); P(_v); P('d');
        return *this;
    }
    w& operator=(w&& x) { _v = move(x._v); P(_v); P('e'); return *this; }

    operator int() const { return _v; }

    template <typename U>
    void operator()(U&& u) { P('f'); forward<U>(u)(*this); ++_v; }

    void operator()(w u) { ++_v; [_p = *this](auto&& p){ P('g'); P(_p); P(p); }(move(u)); }
};

w x{1};

int main(){
    x([=](auto p){ P(x); P(p); x(move(x)); });
}
```

#10

```
#include <algorithm>
#include <iostream>

using namespace std;

template <class F>
void do_it(F f, F l) {
    auto m = f + distance(f, l) / 2;
    if (m == f) return;
    do_it(f, m);
    do_it(m, l);
    rotate(f, m, l);
}

int main() {
    int a[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8 };
    do_it(begin(a), end(a));
    for (auto const& e : a) cout << e;
}
```

+2 bonus points, if you can name the difference to the standard equivalent

#11

```
#include <functional>
#include <iostream>
#include <tuple>

using namespace std;

template <typename T> void P(T const& x) { cout << x; }

auto ops = make_tuple(plus{}, multiplies{});

int main() {
    auto da = [](auto self, auto arg, auto... args) {
        if constexpr (sizeof...(args) == 1)
            return get<sizeof...(args)%2>(ops)(arg, get<0>(make_tuple(args...)));
        else return get<sizeof...(args)%2>(ops)(arg, self(self, args...));
    };
    P(da(da, 1, 1, 2, 3, 3, 5, 2));
}
```

Answers

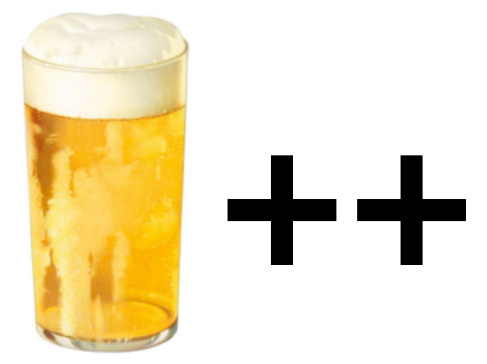
```
#include <iostream>
using namespace std;

template <typename T> void P(T const& x) { cout << x; }

int main() {
    int a[]{1,2,3,4};
    P(0);
    for (auto x : a)
        P(x);
    P(6);
}
```

```
012346
```

+1 bonus if it was discussed in your group, that x is taken by value



C++ Pub Quiz

Sponsored by Bloomberg

#	Answer	Notes	Score	Bonus
0	12346	auto x : a, copy by value		
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Team Name: Team Marvin

Start Bonus	Score	Bonus	Total
10			

accu 2019

++

C++ Pub Quiz

Sponsored by Bloomberg

#	Answer	Notes	Score	Bonus
0	12346	auto x : a, copy by value	3	1
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Team Name: Team Marvin

Start Bonus	Score	Bonus	Total
10			

accu 2019

#1

```
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

int main() {
    vector<pair<int,char>> v = { {1,'b'}, {2,'a'}, {1,'c'}, {2,'b'}};
    stable_sort(begin(v), end(v), [](auto const& x, auto const& y)
        return y.first == min(x.first, y.first);
    });
    for (auto x : v) { cout << x.second; }
    stable_sort(begi
        return y.fir
    });
    for (auto x : v)
}
```

bacbbcab

+1 bonus if you have discussed, that even Alex Stepanov says that std::max is wrong, because it returns the first argument when both arguments are equivalent, but it should return the second

#2

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

template <typename... T> auto P(T... x) { (cout << ... << x); return get

template <typename T>
struct w {
    T _v;
    w(T v): _v(v) {}
    operator T() const { return _v; }
};

template <typename T, typename U>
int operator,(T a, U b) { return P(a + b); }

template <t
Y f(X x, Y
    P('x')
    return
}

int a{1}, c{2};
w b{3}, d{4};

int main() {
    P(f(a, b, P(c,d)));
}
```

```
24x132y55355
```

Since you should never overload the comma operator, there is no bonus

#3

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

template <typename T>
void P(const T& x) { cout << x; }

struct foo {
    int _v;
    foo(int v = 0) : _v(v) {}
    auto bar() & { return _v; }
    auto bar() const& { return _v+1; }
    auto bar() && { return _v+2; }

    auto operator|(int v) & { return foo(_v+v); }
    auto operator|(int v) const& { return
foo(_v+v); }
    auto operator

    auto operator
    auto operator
        return ba
    }
    auto operator
        return st
    }

    operator int() const { return _v; }
};
```

```
int main() {
    foo a1;
    P(a1.bar());
    const foo a2;
    P(a2.bar());
    P(foo().bar());
    P(a1 | 2);
    P(a2 | 3);
    P(a1 & 2);
    P(a2 & 3);
```

```
01223245103
```

+1 bonus if it was discussed in your group, that *this is an lvalue within an && method

#4

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

template <typename T> void P(T const& x) { cout << x; }

template <typename...T>
auto s1(T... t) { return (t + ... ); }

template <typename...T>
auto s2(T... t) { return (... *= t); }

template <typename...T>
auto s3(T... t) { return (P(t), ... , P(1)), 1; }

template <typenar
auto s4(T... t)

int main() {
    P(s1(1,2,3))
    P(s2(4,5,6));
    P(s3(7,8,9));
    P(s4(1,2,4));
}
```

612078910

+1 bonus if in your discussion the terms “unary left / right fold” were dropped

+1 bonus if in your discussion the term “binary left / right fold” were dropped

accu 2019

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

void P(int x) { cout << x; }

template <size_t I>
struct parent
{
  size_t value = I;
};

template <size_t... I>
struct child : parent<I>...
{
    operator int() const { return ( this->parent<I>::value + ...); }

    template <size_t...J, typename... T>
    void set(index_sequence<J...>, T... t) {
        P(*initializer_list<int>{(this->parent<I-J>::value *= t, 0)...}.begin());
    }

    template <typename
    void set(T... t) {
        set(make_index
    }
};

int main() {
    child<1,2,3> b;
    P(b);
    b.set(4,5,6);
    P(b);
    b.set(7,8,9);
    P(b);
}
```

601250604 85

#5

+1 bonus if it was discussed in your group the term variadic base class

+1 bonus if you sent curses from your group to the author of the code snipet.

#6

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

template <typename T> void P(T const& x) { cout << x; }

template<typename P>
void foo(string const& str, P& p) {
    for (auto ch : str)
        p(ch);
}

int main() {
    int num = 1;
    string str("abc");
    auto f
    foo(st
    foo(st
    P(num)
}
```

```
1a2b3c4a5b6c1
```

+2 bonus if the C++ Pub Quiz 2016 was mentioned

#7

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

template<char S, size_t C>
struct R {
    string d;
    size_t c = 0;

    struct E {
        size_t& c;
        bool operator()(string::iterator it) const {
          if (*it == S) ++c;
          return c != C;
        }
    };

    auto b
    auto e

    frienc
};

int main() {
    auto t = "12012300123450001234561234567";
    for (auto c : R<'0', 1>{t}) cout << c;
    for (auto c : R<'1', 2>{t}) cout << c;
    for (auto c : R<'2', 3>{t}) cout << c;
}
```

1212012012300 1

+1 bonus if the term Sentinel was discussed in your group.

+1 bonus if it was mentioned that this feature was added with c++17

accu 2019

```cpp
#include <algorithm>
#include <iostream>
#include <iterator>
#include <numeric>

using namespace std;

int main() {
    int a[] = {2, 3, 4, 5, 2, 1, 5};
    nth_element(begin(a), &a[4], end(a));
    cout << a[5] << reduce(begin(a), &a[5]);
}
```

+1 when it was mentioned in your group, that nth_element was part of one of Sean Parent's better code talks.

```
512
```

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

template <typename T> void P(T const& v) { cout << v; }

template <typename T>
struct w
{
    T _v;
    w(T v) : _v(move(v)) { P('a'); }
    w(w const& x): _v(x._v) { P(_v); P('b'); }
    w(w&& x) { *this = move(x); P(_v); P('c'); }
    w& operator=(w const& x) {
        auto tmp = x; *this = move(tmp); P(_v); P('d');
        return *this;
    }
    w& operator=(w&& x) { _v = move(x._v); P(_v); P('e'); return *this; }

    operator int() const { return _v; }
```

af1b111e1c2bg21

```cpp
}

w

int main(){
    x([=](auto p){ P(x); P(p); x(move(x)); });
}
```

#9

+1 bonus point, if you have discussed the different kind of capturing. *this was added with C++17

+1 bonus point, if you asked yourself if there is a superlative to crap

accu 2019

```
#include <algorithm>
#include <iostream>

using namespace std;

template <class F>
void do_it(F f, F l) {
    auto m = f + distance(f, l) / 2;
    if (m == f) return;
    do_it(f, m);
    do_it(m, l);
    rotate(f, m, l);
}

int main() {
    int a[] = { 0,
    do_it(begin(a)
    for (auto cons
}
```

#10

+2 bonus point, if you saw, that this reverse just needs forward iterators and the standard needs bidirectional iterators

```
876543210
```

accu 2019

#11

```
#include <functional>
#include <iostream>
#include <tuple>

using namespace std;

template <typename T> void P(T const& x) { cout << x; }

auto ops = make_tuple(plus{}, multiplies{})

int main() {
    auto foo = [](auto self, auto          . args) {
        if constexpr (sizeof...(a
            return get<sizeof...(          s)(arg,
get<0>(make_tuple(args...)));
        else return get<sizeof...          ps)(arg, self(self, args...));
    };
    P(foo(foo, 1, 1, 2, 3, 3, 5, 2));
}
```

42

+1 bonus if you discussed that plus<T>, multiplies<T>, and others are defaulted to void since C++14 and so they are easier to use.

+1 bonus if in your discussion was mentioned the recursive reduction of the lambda

+1 bonus if in your group words like “crap”, “who wrote that?”, “what the heck…” or worse where mentioned.

C++ Pub Quiz

++

Sponsored by
Bloomberg

#	Answer	Notes	Score	Bonus
0	012346	auto x : a, copy by value	3	1
1	bacbbcab	std::max UB (mistake in puzzle)	3	1
2	24x32y55355	-	3	0
3	01223245103	&& method, *this is an l-value	3	1
4	6120789110	unary left&right, binary left&right	3	2
5	60125060485	variadic base class, curses to the author	3	2
6	1a2b3c4a5b6c1	C++ pub quiz 2016	3	2
7	12120120123001	Sentinel, new C++17 feature	3	2
8	512	Sean Parent's better code talks	3	1
9	af1b111b2b1bg21	lambda capture *this, C++17 feature	3	2
10	876543210	only forward iterator, standard requires indirect.	3	2
11	42	default to void, recursive lambda, wtf	3	3

Team Name: Team Marvin

Start Bonus	Score	Bonus	Total
10	36	19	65

accu 2019

++

If you are into C++ you should definitely visit:
isocpp.org

If you enjoy C++ quiz in general, then have a go at:
cppquiz.org

If you like the ACCU conference, remember the date
2020-03-24 to 2020-03-28 for ACCU 2020

And finally, if you are curious about the sponsor for this particular event:
techatbloomberg.com