Started on	Thursday, 2 February 2023, 8:55 PM
State	Finished
Completed on	Thursday, 2 February 2023, 9:47 PM
Time taken	52 mins 3 secs

Information

Read Section 6.5 of the textbook and the lecture handout on Default Function Parameters.

Assume all necessary headers are included. Don't provide "doesn't compile" as a valid answer only because certain headers are NOT included in a code fragment.

When it's required to compile a code fragment, add all necessary headers and compile/link using the following compilation options:

```
-std=c++17 -pedantic-errors -Wall -Wextra -Werror
```

Question **1**
Correct
Marked out of 3.00

Given the declaration

```
void mystery(int x, int y = 10, double d = 11.2);
```

write the comma-separated initializers for parameters `x`, `y`, and `z` [in this order] when expression `mystery(18.9,6.5,13.2)` is evaluated. Write **CTE** if the expression doesn't compile. Your answer should not contain any extraneous whitespace characters.

Answer: 18,6,13.2 ✓

Question **2**
Correct
Marked out of 6.00

Which of the following can be used as a default function argument?

Select one or more:

- [✓] function call (as long as the function being called is declared before its use in the declaration) ✓
- [✓] heap variables, as for example `void foo(int a= *(new int));` ✓
- [ ] local variables
- [✓] variables declared in anonymous namespaces ✓
- [✓] literals ✓
- [✓] global variables ✓

Question **3**

Correct

Marked out of 5.00

Given the following code fragment:

```
void secret(int x, int y, double t, char z = 'A', int n = 67, char v = 'G', double w = 78.34);

int a, b;
char ch;
double d;

// now, we call function secret ...
```

which of the following calls to function `secret` evaluate without requiring implicit conversions by the compiler?

Select one or more:

- [✓] `secret(a, 15, 34.6, 'B', 87, ch);` ✓
- [ ] `secret(b, 25, 48.76, 'D', 4567, 78.34);`
- [ ] `secret(a, 15, 34.6, 46.7);`
- [✓] `secret(a, b, d);` ✓
- [✓] `secret(b, a, 14.56, 'D');` ✓

Question **4**

Correct

Marked out of 3.00

Given the following declaration

```
void mystery(int x, int y = 10, double d = 11.2);
```

write the comma-separated initializers for parameters `x`, `y`, and `z` [in this order] when expression `mystery(5)` is evaluated. If the expression doesn't compile, write **CTE**. Your answer should not contain any extraneous whitespace characters.

Answer: 5,10,11.2 ✓

Question **5**

Correct

Marked out of 7.00

You have used function pointers in programs. Now, consider the combination of function pointers and arrays in the following code fragment:

```
int f1(); // assume f1 returns 16
int f2(); // assume f2 returns 32
int f3(); // assume f3 returns 64

int (*PF[])() {f1, f2, f3};
```

Now, evaluate the following expressions.

`f3`	Address of function f3()	✓
`PF[1]()`	32	✓
`PF()`	Expression doesn't compile	✓
`(*(PF+1))`	Address of function f2()	✓
`(*PF+1)()`	Expression doesn't compile	✓
`(*(PF+1))()`	32	✓
`(**(PF+1))()`	32	✓

Question **6**
Correct
Marked out of 7.00

To know the types of C++ literals, review Table 2.2 under Section 2.1.3 of the text.

`3.145f`	float	✓
`6`	int	✓
`43543534535ull`	unsigned long long	✓
`4354353453511`	long long	✓
`10.888`	double	✓
`788093L`	long	✓
`788093.55L`	long double	✓

Question **7**
Correct
Marked out of 9.00

Make sure to know the sizes of fundamental types on a 64-bit machine with a 64-bit compiler such as g++.

`sizeof(long long)`	8	✓
`sizeof(long double)`	16	✓
`sizeof(float)`	4	✓
`sizeof(double)`	8	✓
`sizeof(bool)`	1	✓
`sizeof(short)`	2	✓
`sizeof(long)`	8	✓
`sizeof(char)`	1	✓
`sizeof(int)`	4	✓

Question **8**
Correct
Marked out of 6.00

Given the following C++ code fragment:

```
void foo(int = 4, double = 6.1);

foo();    // Line 1
foo(7);   // Line 2
foo(9.2); // Line 3
```

rewrite each statement [labeled in comments] as an equivalent explicit function call statement [including the semicolon]. Don't use any whitespace in your answer.

Line 1	foo(4,6.1);	✓
Line 2	foo(7,6.1);	✓
Line 3	foo(9.2,6.1);	✓

Question **9**

Correct

Marked out of 11.00

Consider the following function declarations in a C++ code fragment:

```
void yoo();                        // Line 1
void yoo();                        // Line 2
void bar(int, double=5.0);         // Line 3
void bar(int = 4, double);         // Line 4
void baz(double=6.77, short);      // Line 5
void baz(double, short=5);         // Line 6
void goo(char x=' ', short='h');   // Line 7
void goo(char y='n', short=50);    // Line 8
void hoo(float, short, float=6.0f); // Line 9
void hoo(float=8, short, float);   // Line 10
void hoo(float, short=8, float);   // Line 11
```

Determine whether the function declarations [identified by line numbers] are legal or illegal.

Line 11	Legal	✓
Line 10	Illegal because of missing default arguments on right-most parameters	✓
Line 3	Legal	✓
Line 2	Legal	✓
Line 9	Legal	✓
Line 4	Legal	✓
Line 5	Illegal because of missing default argument on parameters.	✓
Line 1	Legal	✓
Line 7	Legal	✓
Line 6	Legal	✓
Line 8	Illegal because default arguments are being re-specified	✓

Question **10**

Correct

Marked out of 4.00

Given the declaration

```
void mystery(int a, int b = 10, char z = '*');
```

which of the following calls to function `mystery` are legal?

Select one or more:

- [x] `mystery(0, 0, '*');` ✓
- [x] `mystery(5);` ✓
- [x] `mystery(5, 8);` ✓
- [x] `mystery(6, '#');` ✓

Question **11**

Correct

Marked out of 6.00

Given the following definition

```
int mystery(int a, double b, char ch) {
  return ('A' <= ch && ch <= 'R') ? (2*a+static_cast<int>(b)) : (static_cast<int>(2*b)-a);
}
```

write the comma-separated results of evaluating the following three expressions: `mystery(5,4.3,'B')`, `mystery(4,9.7,'v')`, `mystery(6,3.9,'D')` [in that order]. Your answer should not contain any extraneous whitespace characters.

Answer: 14,15,15 ✓

Question **12**
Correct
Marked out of 3.00

Which of the following function declarations illegal?

Select one or more:

- [ ] `void foo(int a, int b, int c);`
- [x] `int foo(int a, int b, int c=a);` ✓
- [x] `void foo(int a, int b = 3, int c);` ✓

Question **13**
Correct
Marked out of 2.00

Given the following definitions

```
unsigned int ui = 8;
int i = -9;
```

does the expression `ui+i` evaluate to value `-1`?

Select one:

True [✓ False] ✓

Question **14**
Correct
Marked out of 3.00

Given the declaration

```
void mystery(int x, int y = 10, double d = 11.2);
```

write the comma-separated initializers for parameters `x`, `y`, and `z` [in this order] when expression `mystery(5, 8.9)` is evaluated. If the expression doesn't compile, write **CTE**. Your answer should not contain any extraneous whitespace characters.

Answer: 5,8,11.2 ✓

Question **15**
Correct
Marked out of 6.00

Given the definition

```
int secret(int u, int v = 5, double z = 3.2) {
  u = static_cast<int>(2*v + z) + u;
  return u+v*z;
}
```

write the comma-separated values that expressions `secret(6)`, `secret(3,4)`, `secret(3,0,2.8)` evaluate to [in that order]. Your answer must not contain extraneous whitespace characters.

Answer: 35,26,5 ✓

Question **16**
Correct
Marked out of 4.00

Given the following definition:

```
double secret(double x, int y, std::string name) {
    // some hush-hush stuff we don't care about ...
}
```

which of the following are legal declarations of function `secret`?

Select one or more:

- [x] `double secret(double x, int y, std::string s);` ✓
- [ ] `secret(double x, int y, std::string s);`
- [ ] `double secret();`
- [x] `double secret(double, int, std::string);` ✓

Question **17**
Correct
Marked out of 3.00

Given the declaration

```
void mystery(int x, int y = 10, double d = 11.2);
```

write the comma-separated initializers for parameters `x`, `y`, and `z` [in this order] when expression `mystery(18.9)` is evaluated. If the expression doesn't compile, write **CTE**. Your answer should not contain any extraneous whitespace characters.

Answer: 18,10,11.2 ✓

Question **18**
Correct
Marked out of 12.00

In C++, some types are related to each other. When two types are related, we can use an object or value of one type where an operand of the related type is expected. Two types are related if there is a conversion between them. In expressions involving operands of different built-in types, C++ defines a set of conversions to transform the operands to a common type. These conversions are carried out automatically without programmer intervention - and sometimes without programmer knowledge. For that reason, they are referred to as ***implicit conversions***. Review your understanding of implicit conversions by reading Section 4.11 of the textbook.

Match the kind of implicit conversion occurring in each expression's evaluation assuming the following definitions:

```
bool b;
char c;
short s;
unsigned short us;
int i;
unsigned int ui;
long l;
unsigned long ul;
float f;
double d;
```

`c + i`	char promoted into integer	✓
`l + c`	char converted into long	✓
`ui + us`	unsigned short promoted into unsigned int	✓
`s = c`	char converted into short	✓
`i + d`	integer converted into double	✓
`s + i`	short promoted into integer	✓
`s = b`	bool converted into short	✓
`d + f`	float promoted into double	✓
`d + s`	short converted into double	✓
`i + b`	bool promoted into int	✓
`ui + s`	short promoted into integer, integer converted into unsigned int	✓
`i + l`	integer converted into long	✓

◄ Quiz 1: Introduction to C++ Jump to... Quiz 3: Review of C++ Namespaces ►