Ray Mitchell, U99999999
MeanOldTeacher@MeanOldTeacher.com
C/C++ Programming I
Section 162461, Ray Mitchell
June 25, 2019
C1A4E0_Quiz.txt
Quiz Answers

1. B
2. D
3. B
4. A
5. B
6. C

C1A4E0 Explanations

In addition to the course book references cited below, these topics are also covered in the live lectures (in-class students) and the recorded lectures (online students).

Function Prototypes (notes 5.2 through 5.5)
If a function is called without a prototype present its arguments are subject to "the usual function argument conversions" and code will be generated to place them into the function's parameters according to those conversion rules. This is not a problem if the converted types are those expected by the function, but the function's parameters will receive garbage values if not. Similarly, if there is no prototype every call to the function will assume that it is returning type **int**. This is not a problem if it actually does return type **int**, but will result in garbage return values if not.

1. **B** See the "Function Prototypes" information above.

2. **D** See the "Function Prototypes" information above.

3. **B** Note 5.8; Only that prototype has the same number of parameters as the function call has arguments. The types of all three arguments will be implicitly converted to the types of the corresponding parameters and the type **long** return type will be implicitly converted to the type **double** assignment variable. The return type is not involved in overload resolution.

4. **A** Notes 5.15, 5.16; Good programming practice dictates that the scope of any item be as small as possible to prevent its inadvertent interaction with other portions of the code. External variables, functions, and items placed in header files are potentially available to everything else in the program. While this is often desirable for header file items, it is often not desirable for functions and external variables. As a result, such functions and external variables should be declared **static** to limit their scope to only the file in which they are declared.

5. **B** Notes 2.13, 5.18; To prevent unexpected operator precedence problems between macros and the code in which they are used, parentheses should be placed around every replacement list that contains more than one token. In addition, if a macro has arguments that are used in its replacement list, parentheses should be placed around each instance of those arguments in the replacement list. Besides that, the most run-time efficient way of accomplishing a task should always be chosen unless there is a compelling reason not to do so, and negation using the unary minus operator is the simplest way to produce negation. Only the indicated answer meets all of these criteria.

6. **C** Notes 2.13, 5.18; The macro replacement process is most easily understood by viewing it as a "copy and paste". That is, each time the preprocessor encounters the use of a macro name it in effect copies the entirety of that macro's replacement list onto the clipboard unaltered and pastes it in place of that macro name. The only difference between this and a true "copy and paste" is that an additional space is placed on end(s) of what gets pasted if necessary to avoid creating any new tokens from something adjacent to the replaced macro name. Assuming the conditions presented in this quiz question, the following is the substitution sequence:

```
#define Quotient(n, d) n / d     macro for Quotient
5 + Quotient(3.5 + 1, 2)         original expression that uses macro Quotient
5 + 3.5 + 1 / 2                  3.5 + 1 is substituted for n and 2 is substituted for d
equals 8.5
```

Note that division has the highest precedence and since 1 and 2 are both integer types 1 divided by 2 equals 0.

```c
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 162461, Ray Mitchell
// June 25, 2019
// C1A4E1_main.c
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function main, which prompts the user for two values,
// calls functions that determine their maximum and minimum, then displays the
// results.
//

#include <stdio.h>
#include <stdlib.h>

// Prototypes for custom functions called by main.
double ComputeMaximum(double val1, double val2);
double ComputeMinimum(double val1, double val2);

//
// Test functions ComputeMinimum and ComputeMaximum by calling them with values
// obtained from user input and display the results.  These functions return the
// minimum and maximum of the two values passed to them as arguments.
//
int main(void)
{
   double value1, value2;

   // Get two user input values and determine which is lesser/greater.
   printf("Enter two space-separated decimal values: ");
   scanf("%lg %lg", &value1, &value2);

   printf("ComputeMinimum(%g, %g) returned %g\n", value1, value2,
      ComputeMinimum(value1, value2));
   printf("ComputeMaximum(%g, %g) returned %g\n", value1, value2,
      ComputeMaximum(value1, value2));

   return EXIT_SUCCESS;
}
```

```
1   //
2   // Ray Mitchell, U99999999
3   // MeanOldTeacher@MeanOldTeacher.com
4   // C/C++ Programming I
5   // Section 162461, Ray Mitchell
6   // June 25, 2019
7   // C1A4E1_ComputeMaximum.c
8   // Windows 10 Professional
9   // Visual Studio 2019 Professional
10  //
11  // This file contains function ComputeMaximum, which returns the maximum of its
12  // two parameter values.
13  //
14
15  //
16  // Determine the maximum of the two parameter values and return it. No special
17  // test is done for the values being equal.
18  //
19  double ComputeMaximum(double val1, double val2)
20  {
21     // Compare <val1> and <val2> and return the maximum.
22     return(val1 > val2 ? val1 : val2);
23  }
```

```
1   //
2   // Ray Mitchell, U99999999
3   // MeanOldTeacher@MeanOldTeacher.com
4   // C/C++ Programming I
5   // Section 162461, Ray Mitchell
6   // June 25, 2019
7   // C1A4E1_ComputeMinimum.c
8   // Windows 10 Professional
9   // Visual Studio 2019 Professional
10  //
11  // This file contains function ComputeMinimum, which returns the minimum of its
12  // two parameter values.
13  //
14
15  //
16  // Determine the minimum of the two parameter values and return it. No special
17  // test is done for the values being equal.
18  //
19  double ComputeMinimum(double val1, double val2)
20  {
21     // Compare <val1> and <val2> and return the minimum.
22     return(val1 < val2 ? val1 : val2);
23  }
```

```cpp
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 162461, Ray Mitchell
// June 25, 2019
// C1A4E2_main.cpp
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function main, which gets input from the user and passes
// it to four overloaded PrintLines functions as appropriate.  These functions
// display zero or more characters zero or more times on zero or more lines as
// determined by the arguments passed to them.
//

#include <iostream>

const int TESTS = 2;     // number of times to execute body of test loop

// Prototypes for custom overloaded functions called by main.
void PrintLines(int charValue, int charCount, int lineCount);
void PrintLines(int charValue, int charCount);
void PrintLines(int charValue);
void PrintLines();

//
// Get user input and pass it to overloaded PrintLines functions to validate
// their operation.
//
int main()
{
   //
   // Each execution of the body of the loop constitutes one set of tests.  Each
   // set gets new user input and passes it as appropriate to call all four
   // versions of the overloaded PrintLines functions.
   //
   for (int testNo = 0; testNo < TESTS; ++testNo)
   {
      char charValue;
      int charCount, lineCount;

      // Get test values from user input.
      std::cout <<
         "Enter a space-separated character, character count, and line count: ";
      std::cin >> charValue >> charCount >> lineCount;

      //
      // Call all four versions of the overloaded PrintLines function with user
      // input values as arguments.  The number of arguments provided determines
      // the version of the function called.
      //
      PrintLines(charValue, charCount, lineCount);
      PrintLines(charValue, charCount);
      PrintLines(charValue);
      PrintLines();
   }
   return 0;
}
```

```
1   //
2   // Ray Mitchell, U99999999
3   // MeanOldTeacher@MeanOldTeacher.com
4   // C/C++ Programming I
5   // Section 162461, Ray Mitchell
6   // June 25, 2019
7   // C1A4E2_PrintLines-0.cpp
8   // Windows 10 Professional
9   // Visual Studio 2019 Professional
10  //
11  // This file contains the 0-parameter version of function PrintLines, which
12  // displays a default character.
13  //
14
15  #include <iostream>
16
17  const char CHAR_TO_PRINT = 'Z';
18
19  //
20  // Print the character in CHAR_TO_PRINT on a line.
21  //
22  void PrintLines()
23  {
24     std::cout << CHAR_TO_PRINT << '\n';
25  }
```

```
1   //
2   // Ray Mitchell, U99999999
3   // MeanOldTeacher@MeanOldTeacher.com
4   // C/C++ Programming I
5   // Section 162461, Ray Mitchell
6   // June 25, 2019
7   // C1A4E2_PrintLines-1.cpp
8   // Windows 10 Professional
9   // Visual Studio 2019 Professional
10  //
11  // This file contains the 1-parameter version of function PrintLines, which
12  // displays the specified character represented by its parameter once.
13  //
14
15  #include <iostream>
16
17  //
18  // Print the character in <charValue> on a line.
19  //
20  void PrintLines(int charValue)
21  {
22     std::cout.put(char(charValue));
23     std::cout << '\n';
24  }
```

```
1   //
2   // Ray Mitchell, U99999999
3   // MeanOldTeacher@MeanOldTeacher.com
4   // C/C++ Programming I
5   // Section 162461, Ray Mitchell
6   // June 25, 2019
7   // C1A4E2_PrintLines-2.cpp
8   // Windows 10 Professional
9   // Visual Studio 2019 Professional
10  //
11  // This file contains the 2-parameter version of function PrintLines, which
12  // displays a specified character a specified number of times, as determined
13  // by its parameters.
14  //
15
16  #include <iostream>
17
18  //
19  // Print the character in <charValue> the number of times specified by
20  // <charCount> on a line.
21  //
22  void PrintLines(int charValue, int charCount)
23  {
24     // This loop counts the number of characters on a line.
25     for (int chars = 0; chars < charCount; ++chars)
26        std::cout.put(char(charValue));
27     std::cout << '\n';
28  }
```

```
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 162461, Ray Mitchell
// June 25, 2019
// C1A4E2_PrintLines-3.cpp
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains the 3-parameter version of function PrintLines, which
// displays a specified character a specified number of times on a specified
// number of lines as determined by its parameters.
//

#include <iostream>

//
// Print the character in <charValue> the number of times specified by
// <charCount> on the number of lines specified by <lineCount>.
//
void PrintLines(int charValue, int charCount, int lineCount)
{
   // This loop counts the number of lines.
   while (lineCount--)
   {
      // This loop counts the number of characters on a line.
      for (int chars = 0; chars < charCount; ++chars)
         std::cout.put(char(charValue));
      std::cout << '\n';
   }
}
```

```
//
// Ray Mitchell, U99999999
// MeanOldTeacher@MeanOldTeacher.com
// C/C++ Programming I
// Section 162461, Ray Mitchell
// June 25, 2019
// C1A4E3_main.cpp
// Windows 10 Professional
// Visual Studio 2019 Professional
//
// This file contains function main, which gets input from the user and passes
// it to default argument function PrintLines as appropriate.  This function
// displays zero or more characters zero or more times on zero or more lines
// as determined by the explicit or default arguments passed to it.
//

#include <iostream>

const int TESTS = 2;     // number of times to execute body of test loop

// Prototype for custom default argument function called by main.
void PrintLines(int charValue = 'Z', int charCount = 1, int lineCount = 1);

//
// Get user input and pass it to default argument function PrintLines to
// validate its operation.
//
int main()
{
   //
   // Each execution of the body of the loop constitutes 1 set of tests.  Each
   // set gets new user input and passes it as appropriate to the PrintLines
   // function to test its default argument behavior.
   //
   for (int testNo = 0; testNo < TESTS; ++testNo)
   {
      char charValue;
      int charCount, lineCount;

      // Get test values from user input.
      std::cout <<
         "Enter a space-separated character, character count, and line count: ";
      std::cin >> charValue >> charCount >> lineCount;

      //
      // Call the PrintLines function with user input values as arguments.
      // Default values are automatically supplied for omitted arguments.
      //
      PrintLines(charValue, charCount, lineCount);
      PrintLines(charValue, charCount);
      PrintLines(charValue);
      PrintLines();
   }
   return 0;
}
```

```
1   //
2   // Ray Mitchell, U99999999
3   // MeanOldTeacher@MeanOldTeacher.com
4   // C/C++ Programming I
5   // Section 162461, Ray Mitchell
6   // June 25, 2019
7   // C1A4E3_PrintLines.cpp
8   // Windows 10 Professional
9   // Visual Studio 2019 Professional
10  //
11  // This file contains the 3-parameter version of function PrintLines, which
12  // displays a specified character a specified number of times on a specified
13  // number of lines as determined by its parameters.
14  //
15
16  #include <iostream>
17
18  //
19  // Print the character in <charValue> the number of times specified by
20  // <charCount> on the number of lines specified by <lineCount>.
21  //
22  void PrintLines(int charValue, int charCount, int lineCount)
23  {
24     // This loop counts the number of lines.
25     while (lineCount--)
26     {
27        // This loop counts the number of characters on a line.
28        for (int chars = 0; chars < charCount; ++chars)
29           std::cout.put(char(charValue));
30        std::cout << '\n';
31     }
32  }
```

```
1   //
2   // Ray Mitchell, U99999999
3   // MeanOldTeacher@MeanOldTeacher.com
4   // C/C++ Programming I
5   // Section 162461, Ray Mitchell
6   // June 25, 2019
7   // C1A4E4_MaxOf.h
8   // Windows 10 Professional
9   // Visual Studio 2019 Professional
10  //
11  // This file contains inline functions fMaxOf2 and fMaxOf3 and macros mMaxOf2
12  // and mMaxOf3.  Each returns the maximum of the argument values passed to it.
13  //
14
15  #ifndef C1A4E4_MAXOF_H
16  #define C1A4E4_MAXOF_H
17
18  //
19  // Determine the maximum of the two parameter values and return it. No special
20  // action is taken if the values are equal; that value is simply returned.  Any
21  // scalar types may be used as arguments as long as they are compatible with
22  // each other.
23  //
24  #define mMaxOf2(xx, yy) (((xx) > (yy)) ? (xx) : (yy))
25
26  //
27  // Determine the maximum of the three parameter values and return it. This is
28  // not done by direct comparison but, rather, by calling macro mMaxOf2 twice to
29  // do the comparisons.  No special action is taken if the values are equal; that
30  // value is simply returned. Any scalar types may be used as arguments as long
31  // as they are compatible with each other.
32  //
33  #define mMaxOf3(xx, yy, zz) (mMaxOf2((xx), mMaxOf2((yy), (zz))))
34
35  //
36  // Determine the maximum of the two parameter values and return it. No special
37  // action is taken if the values are equal; that value is simply returned.
38  //
39  inline long double fMaxOf2(long double xx, long double yy)
40  {
41     // Compare <xx> and <yy> and return the maximum.
42     return(xx > yy ? xx : yy);
43  }
44
45  //
46  // Determine the maximum of the three parameter values and return it. This is
47  // not done by direct comparison but, rather, by calling function fMaxOf2 twice
48  // to do the comparisons.  No special action is taken if the values are equal;
49  // that value is simply returned.
50  //
51  inline long double fMaxOf3(long double xx, long double yy, long double zz)
52  {
53     // Compare <xx>, <yy>, and <zz> and return the maximum.
54     return(fMaxOf2(xx, fMaxOf2(yy, zz)));
55  }
56  #endif
```

```
1   //
2   // Ray Mitchell, U99999999
3   // MeanOldTeacher@MeanOldTeacher.com
4   // C/C++ Programming I
5   // Section 162461, Ray Mitchell
6   // June 25, 2019
7   // C1A4E4_main.cpp
8   // Windows 10 Professional
9   // Visual Studio 2019 Professional
10  //
11  // This file contains function main, whose purpose is to display the return
12  // values of function fMaxOf3 and macro mMaxOf3, both of which return the
13  // maximum of the argument values passed to them.
14  //
15
16  #include <iostream>
17  using std::cin;
18  using std::cout;
19  #include "C1A4E4_MaxOf.h"
20
21  //
22  // Test the mMaxOf3 macro and the fMaxOf3 function using values obtained from
23  // user input.
24  //
25  int main()
26  {
27     long double val1, val2, val3;
28
29     cout << "Enter three space-separated decimal values: ";
30     cin >> val1 >> val2 >> val3;
31
32     cout << "mMaxOf3(" << val1 << ", " << val2 << ", " << val3
33        << ") returned " << mMaxOf3(val1, val2, val3) << '\n';
34     cout << "fMaxOf3(" << val1 << ", " << val2 << ", " << val3
35        << ") returned " << fMaxOf3(val1, val2, val3) << '\n';
36
37     return 0;
38  }
```