Consolidated Assignment 4 Report

This report contains the graded results for the newest of each exercise submitted to the assignment checker prior to 2/9/2022 12:05:59 AM PST.

Student Name: Phillip Ward
Student ID: U09339367
Contact email: phillip.ward@seagate.com
C/C++ Programming I (Section 162461)

Submitted:
- Exercise 0: 2/6/2022 3:37:50 PM PST
- Exercise 1: 2/6/2022 2:00:55 PM PST
- Exercise 2: 2/6/2022 2:11:04 PM PST
- Exercise 3: 2/6/2022 2:53:35 PM PST
- Exercise 4: 2/6/2022 3:28:26 PM PST

Credit to be deducted for uncorrected assignment checker issue(s):
- Exercise 1: 0.8 points (~25%) minimum
- Exercise 2: 1 point (~25%) minimum
- Exercise 3: 0.8 points (~25%) minimum
- Exercise 4: 1 point (~25%) minimum

Score (out of 20 possible): 12.7

Additional deductions:
- Exercise 0: -3
- Exercise 1 requirements not met: -0.7

Phillip,
Because it takes great deal of time to review and personally comment on student exercise submissions, especially those with uncorrected assignment checker issues, my policy is not to do so past assignment 3 for students who choose to leave many such issues remaining. In such cases no code markups will be made and any additional deductions I deem necessary will be only briefly listed as you see above. Specific details on such additional deductions will only be provided upon individual request.
Ray Mitchell

THIS WAS SENT FROM A NOTIFICATION-ONLY ADDRESS THAT CANNOT ACCEPT INCOMING MAIL.
For help please contact the instructor at the email address provided on the "Home" page of the course's Canvas website. The assignment checker DOES NOT GRADE your submissions but merely reports on issues so you can avoid credit loss by making corrections and resubmitting. ALL GRADING IS DONE MANUALLY BY THE INSTRUCTOR after the assignment deadline based solely upon the NEWEST submission of each exercise that was submitted BEFORE THE ASSIGNMENT DEADLINE. NO CREDIT will be given for anything submitted after the deadline.

From: Phillip Ward <mailto:phillip.ward@seagate.com>
Subject: C1A4E0_162461_U09339367
Submitted: 2/6/2022 3:37:50 PM PST
Course: C/C++ Programming I (Section 162461)
Student's name: Phillip Ward
Contact email: phillip.ward@seagate.com
Student ID: U09339367
Assignment 4, Exercise 0 (003885753M01005X29885)
Exercise point value: 6
File submitted:
C1A4E0_Quiz.txt

NOTE: The assignment checker does not check the correctness of answers for this exercise.

Your submission has been accepted and will be graded manually by the instructor. You may resubmit it as many times as you wish BEFORE THE ASSIGNMENT DEADLINE. NO CREDIT will be given for anything submitted after the deadline.

-3

Phillip Ward U09339367
Phillip.Ward@seagate.com
C/C++ Programming I
162461 Ray Mitchell
02/06/2022
C1A4E0_Quiz.txt
Quiz Answers

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

THIS WAS SENT FROM A NOTIFICATION-ONLY ADDRESS THAT CANNOT ACCEPT INCOMING MAIL.
For help please contact the instructor at the email address provided on the "Home" page of the course's Canvas website. The assignment checker DOES NOT GRADE your submissions but merely reports on issues so you can avoid credit loss by making corrections and resubmitting. ALL GRADING IS DONE MANUALLY BY THE INSTRUCTOR after the assignment deadline based solely upon the NEWEST submission of each exercise that was submitted BEFORE THE ASSIGNMENT DEADLINE. NO CREDIT will be given for anything submitted after the deadline.

From: Phillip Ward <mailto:phillip.ward@seagate.com>
Subject: C1A4E1_162461_U09339367
Submitted: 2/6/2022 2:00:55 PM PST
Course: C/C++ Programming I (Section 162461)
Student's name: Phillip Ward
Contact email: phillip.ward@seagate.com
Student ID: U09339367
Assignment 4, Exercise 1 (001701103M01005X54701)
Exercise point value: 3
Files submitted:
C1A4E1_ComputeMaximum.c
C1A4E1_ComputeMinimum.c
C1A4E1_ComputeMinMax.h
C1A4E1_main.c

"Static analysis" results:
6 warnings as follows:
2 exercise-specific warnings (custom validator);
4 poor practice warnings (custom validator);
3 advisories as follows:
2 inter-token spacing advisories (custom validator);
1 miscellaneous advisory (custom validator);
3 recommendations;
"Runtime" results:
Program ran - No errors detected during preliminary testing (SEE ATTACHMENT);

STANDARD GRADING POLICY:
The MINIMUM deduction is the greater of the following for static analysis issues plus a possible additional deduction for runtime issues, if any:
100% if any "goto" statement is used, else
~45% if any compiler or linker error, else
~25% if any warning, else
~15% if any advisory, else
0% if any recommendation.

C1A4E1: YOUR MINIMUM DEDUCTION: 0.8 points (~25%) To avoid deductions please correct this exercise and resubmit to the assignment checker before the assignment deadline.

The custom validator found 12 problems.
(http://www.MeanOldTeacher.com/AssignmentCheckerKnownIssues.pdf)

??????????
warning W701: Exercise requirement violation
*** EXPLANATION ***
Required function "ComputeMinimum" was not found.

??????????
warning W701: Exercise requirement violation
*** EXPLANATION ***

```
Required function "ComputeMaximum" was not found.

??????????
C1A4E1_ComputeMaximum.c(...) advisory A204:  1 unwanted space as follows:
   Line 15, column 11 (between 'n' and '(')

??????????
C1A4E1_ComputeMinimum.c(...) advisory A204:  1 unwanted space as follows:
   Line 15, column 11 (between 'n' and '(')

??????????
C1A4E1_ComputeMinMax.h(...) warning W528:  4 meaningless/cryptic identifiers as
follows:
   Line 15, column 26:  a
   Line 15, column 36:  b
   Line 16, column 26:  a
   Line 16, column 36:  b
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc.
should convey the meaning but not the value of what they represent.  For example, to
count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no
meaning, whereas names like "studentNumber" or "responseCount" provide some insight
into their purposes.  Although there are some cases in which less meaningful names may
be appropriate, such as when a variable is being used for multiple unrelated purposes
or when names like "x", "y", "z", etc. are used to represent coordinates or abstract
equation variables, this is usually not the situation for exercises in this course.
However, if you believe that your name choice is appropriate in this case, please
contact the instructor to discuss it.

??????????
C1A4E1_main.c(28) recommendation R154:  Use of unnecessary conversion specification
"%.1lf" in "printf":
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
The letter "ell" after the percent sign in the %le, %lf, and %lg conversion
specifications when used in the printf family of functions serves absolutely no purpose
other than to clutter code.  These verbose specifiers work identically to %e, %f, and
%g for both type float and type double arguments and were added to the language
standards primarily because of programmer misuse.  %e, %f, and %g are recommended
instead for both type float and type double in the printf family (but not in the scanf
family, where they are different).

??????????
C1A4E1_main.c(29) recommendation R154:  Use of unnecessary conversion specification
"%.1lf" in "printf":
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
The letter "ell" after the percent sign in the %le, %lf, and %lg conversion
specifications when used in the printf family of functions serves absolutely no purpose
other than to clutter code.  These verbose specifiers work identically to %e, %f, and
%g for both type float and type double arguments and were added to the language
standards primarily because of programmer misuse.  %e, %f, and %g are recommended
instead for both type float and type double in the printf family (but not in the scanf
family, where they are different).

??????????
C1A4E1_main.c(...) recommendation R151:  "Landscape" page orientation detected
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
Of the 32 total lines in your file, 2 of them (28 and 29) exceed the 80 "portrait"
```

orientation column limit. As a result "landscape" orientation has been selected instead, which allows 110 columns. However, portrait is more common and is usually preferred because shorter lines are more readable. Long lines are often caused by unnecessarily long identifier names, which may or may not be a problem in your code. The recommended 80-column limit can be accidentally exceeded if an editor does not provide a clear indication of which columns characters are in, but when configured properly any good code editor will provide it by either displaying column numbers directly, displaying a vertical column "guideline", highlighting characters that go past a specified column, etc. Please consult your editor's documentation to see if it can provide a convenient column number indication to help with your code formatting. If it cannot, I recommend using one that can.

??????????
C1A4E1_main.c(...) advisory A206: Unwanted blank line as follows:
   Line 17
*** EXPLANATION ***
Although thoughtfully placed blank lines can make code more readable, excessive or inappropriately placed blank lines only reduce readablity.

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/04/2022
7  // C1A4E1_ComputeMinMax.h
8  // Win10
9  // g++ 11.2.0
10 //
11 // Header file for compute min and max functions
12 //
13 #ifndef C1A4E1_COMPUTEMINMAX_H
14 #define C1A4E1_COMPUTEMINMAX_H
15 double ComputeMax(double a, double b);
16 double ComputeMin(double a, double b);
17 #endif
```

1 80 81 <—— Not Recommended ——> 110

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/04/2022
7  // C1A4E1_main.c
8  // Win10
9  // g++ 11.2.0
10 //
11 // A program that outputs the max and min between two numbers
12 //
13 #include <stdio.h>
14 #include "C1A4E1_ComputeMinMax.h"
15
16 int main(void) {
17
18     //Get input
19     double input1, input2;
20     printf("input two decimal numbers separated by a space:\n");
21     scanf("%lf %lf", &input1, &input2);
22
23     //Do computation
24     double minVal = ComputeMin(input1, input2);
25     double maxVal = ComputeMax(input1, input2);
26
27     //Print Results
28     printf("ComputeMinimum(%.1lf, %.1lf) returned %.1lf\n", input1, input2, minVal);
29     printf("ComputeMaximum(%.1lf, %.1lf) returned %.1lf\n", input1, input2, maxVal);
30
31     return(0);
32 }
```

```
1
//
// Phillip Ward U09339367
// Phillip.Ward@seagate.com
// C/C++ Programming I
// 162461 Ray Mitchell
// 02/04/2022
// C1A4E1_ComputeMaximum.c
// Win10
// g++ 11.2.0
//
// Contains a function to return the maximum between two values
//

double ComputeMax(double a, double b) {
    return ((a > b) ? a : b);
}
```

```
1
//
// Phillip Ward U09339367
// Phillip.Ward@seagate.com
// C/C++ Programming I
// 162461 Ray Mitchell
// 02/04/2022
// C1A4E1_ComputeMinimum.c
// Win10
// g++ 11.2.0
//
// Contains a function to return the minimum between two values
//

double ComputeMin(double a, double b) {
    return ((a < b) ? a : b);
}
```

```
********** C1 ASSIGNMENT 4 EXERCISE 1 AUTOMATIC PROGRAM RUN RESULTS **********

************* THE RESULTS BELOW HAVE BEEN PARTIALLY CHECKED AND **************
*************   NO ERRORS WERE FOUND.  HOWEVER, THIS DOES NOT   **************
*************  NECESSARILY MEAN THAT THERE ARE NO ERRORS.  THE  **************
*************  INSTRUCTOR WILL DO A MORE THOROUGH CHECK DURING  **************
*************                   MANUAL GRADING.                  **************


------------------------------ START OF 1ST RUN ------------------------------

input two decimal numbers separated by a space:
3 3
ComputeMinimum(3.0, 3.0) returned 3.0
ComputeMaximum(3.0, 3.0) returned 3.0


------------------------------- END OF 1ST RUN -------------------------------

------------------------------ START OF 2ND RUN ------------------------------

input two decimal numbers separated by a space:
-7.98 7.13
ComputeMinimum(-8.0, 7.1) returned -8.0
ComputeMaximum(-8.0, 7.1) returned 7.1


------------------------------- END OF 2ND RUN -------------------------------

------------------------------ START OF 3RD RUN ------------------------------

input two decimal numbers separated by a space:
2000.45 0
ComputeMinimum(2000.5, 0.0) returned 0.0
ComputeMaximum(2000.5, 0.0) returned 2000.5


------------------------------- END OF 3RD RUN -------------------------------

------------------------------ START OF 4TH RUN ------------------------------

input two decimal numbers separated by a space:
54e-2 86e-1
ComputeMinimum(0.5, 8.6) returned 0.5
ComputeMaximum(0.5, 8.6) returned 8.6


------------------------------- END OF 4TH RUN -------------------------------

------------------------------ START OF 5TH RUN ------------------------------

input two decimal numbers separated by a space:
86e-1 54e-2
ComputeMinimum(8.6, 0.5) returned 0.5
ComputeMaximum(8.6, 0.5) returned 8.6


------------------------------- END OF 5TH RUN -------------------------------

------------------------------ START OF 6TH RUN ------------------------------
```

```
input two decimal numbers separated by a space:
-0 0
ComputeMinimum(-0.0, 0.0) returned 0.0
ComputeMaximum(-0.0, 0.0) returned 0.0


------------------------------- END OF 6TH RUN --------------------------------
```

THIS WAS SENT FROM A NOTIFICATION-ONLY ADDRESS THAT CANNOT ACCEPT INCOMING MAIL.
For help please contact the instructor at the email address provided on the "Home" page of the course's Canvas website. The assignment checker DOES NOT GRADE your submissions but merely reports on issues so you can avoid credit loss by making corrections and resubmitting. ALL GRADING IS DONE MANUALLY BY THE INSTRUCTOR after the assignment deadline based solely upon the NEWEST submission of each exercise that was submitted BEFORE THE ASSIGNMENT DEADLINE. NO CREDIT will be given for anything submitted after the deadline.

From: Phillip Ward <mailto:phillip.ward@seagate.com>
Subject: C1A4E2_162461_U09339367
Submitted: 2/6/2022 2:11:04 PM PST
Course: C/C++ Programming I (Section 162461)
Student's name: Phillip Ward
Contact email: phillip.ward@seagate.com
Student ID: U09339367
Assignment 4, Exercise 2 (001353984M01005X66353)
Exercise point value: 4
Files submitted:
C1A4E2_main.cpp
C1A4E2_PrintLines.hpp
C1A4E2_PrintLines-0.cpp
C1A4E2_PrintLines-1.cpp
C1A4E2_PrintLines-2.cpp
C1A4E2_PrintLines-3.cpp

"Static analysis" results:
34 warnings as follows:
5 magic number warnings (custom validator);
29 poor practice warnings (custom validator);
11 advisories as follows:
2 inter-token spacing advisories (custom validator);
9 miscellaneous advisories (custom validator);
2 recommendations;
"Runtime" results:
Program ran - No errors detected during preliminary testing (SEE ATTACHMENT);

STANDARD GRADING POLICY:
The MINIMUM deduction is the greater of the following for static analysis issues plus a possible additional deduction for runtime issues, if any:
100% if any "goto" statement is used, else
~45% if any compiler or linker error, else
~25% if any warning, else
~15% if any advisory, else
0% if any recommendation.

C1A4E2: YOUR MINIMUM DEDUCTION: 1 point (~25%) To avoid deductions please correct this exercise and resubmit to the assignment checker before the assignment deadline.

The custom validator found 47 problems.
(http://www.MeanOldTeacher.com/AssignmentCheckerKnownIssues.pdf)

??????????
C1A4E2_main.cpp(...) warning W740: Scope of one or more variables is too wide in function "main".
*** EXPLANATION ***
Good practice dictates that whenever practical the scope of variables and other program items should be as small as possible to lessen the chance they will be accessed by

parts of the code that should not have access to them. This is a similar philosophy to that used when granting security clearances, that is, the fewer people that have security clearances the less the likelihood of a security breach. In your code the non-constant variable named numIters has been declared prior to the "for" statement that begins on line 22. If this variable is being used to specify the loop count limit it should be declared as "const". Otherwise it should be declared within the body of the loop because, if your code is written correctly, it will only be used there and will not need to retain its value between loop body executions. For more details please see the file named "LimitingTheScopeOfVariables.pdf", which is attached to this email.

??????????
C1A4E2_main.cpp(26) warning W357: Inappropriate "Magic Number" in identifier "input2" on line 26, column 13: 2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment but can also refer to an embedded character literal or string literal, or to an identifier named for the value it represents rather than its purpose. For more details please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to this email.

??????????
C1A4E2_main.cpp(28) warning W357: Inappropriate "Magic Number" in identifier "input2" on line 28, column 26: 2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment but can also refer to an embedded character literal or string literal, or to an identifier named for the value it represents rather than its purpose. For more details please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to this email.

??????????
C1A4E2_main.cpp(30) warning W357: Inappropriate "Magic Number" in identifier "input2" on line 30, column 33: 2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment but can also refer to an embedded character literal or string literal, or to an identifier named for the value it represents rather than its purpose. For more details please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to this email.

??????????
C1A4E2_main.cpp(31) warning W357: Inappropriate "Magic Number" in identifier "input2" on line 31, column 33: 2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment but can also refer to an embedded character literal or string literal, or to an identifier named for the value it represents rather than its purpose. For more details please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to this email.

??????????
C1A4E2_main.cpp(35) recommendation R150: No return statement in function "main".
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
The language standards permit the "main" function to be implemented without a return statement, in which case a value of 0 will be automatically returned if the end of the function's closing brace is reached. However, the most accepted programming practice is to provide an explicit return statement anyway just for consistency with other functions, all of which require an explicit return statement if they are not declared

```
to return void.

??????????
C1A4E2_main.cpp(...) warning W528:  3 meaningless/cryptic identifiers as follows:
   Line 22, column 14:  i
   Line 22, column 21:  i
   Line 22, column 35:  i
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc.
should convey the meaning but not the value of what they represent.  For example, to
count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no
meaning, whereas names like "studentNumber" or "responseCount" provide some insight
into their purposes.  Although there are some cases in which less meaningful names may
be appropriate, such as when a variable is being used for multiple unrelated purposes
or when names like "x", "y", "z", etc. are used to represent coordinates or abstract
equation variables, this is usually not the situation for exercises in this course.
However, if you believe that your name choice is appropriate in this case, please
contact the instructor to discuss it.

??????????
C1A4E2_main.cpp(...) warning W901:  Unnecessary casts as follows:
   Line 30, column 20:  int()
   Line 31, column 20:  int()
   Line 32, column 20:  int()
*** EXPLANATION ***
A cast is typically only necessary for the reasons listed below, but neither should be
present in the code for this particular exercise unless it has been implemented in an
unnecessarily complex manner:
1. An implicit type conversion that causes a warning, OR
2. A desired type conversion that does not occur implicitly.

??????????
C1A4E2_main.cpp(...) warning W350:  1 inappropriate "Magic Number" as follows:
   Line 21, column 20:  2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment
but can also refer to an embedded character literal or string literal, or to an
identifier named for the value it represents rather than its purpose.  For more details
please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to
this email.  NOTE: If you are getting this warning about literal values in an array
initializer list and using indentifiers to represent them is not appropriate, try
declaring the array constant.  If this does not fix the problem or if doing so causes a
compiler error, please contact the instructor to discuss it.

??????????
C1A4E2_main.cpp(...) advisory A237:
  The following title block item is either missing or incorrect:
   File name
*** EXPLANATION ***
Every file must begin with an appropriate title block as illustrated in the course
document titled "How to Prepare and Submit Assignments".

??????????
C1A4E2_main.cpp(...) advisory A206:  Unwanted blank line as follows:
   Line  18
*** EXPLANATION ***
Although thoughtfully placed blank lines can make code more readable, excessive or
inappropriately placed blank lines only reduce readablity.

??????????
```

```
C1A4E2_PrintLines.hpp(2) warning W551:  Improper "Include Guard" naming:
C1A4E2_PRINTLINES
*** EXPLANATION ***
An "Include Guard" (appendix note D.2) must be implemented in every header file (but
never in an implementation file) to prevent multiple inclusions of its contents if it
is included more than once.  The most standard "Include Guard" naming convention is the
uppercase name of the file itself with all non-alphanumeric characters replaced by
underscores and, if the name starts with a number, an underscore prepended.  For your
file:
   C1A4E2_PrintLines.hpp
the "Include Guard" name must be (or at least must end with):
   C1A4E2_PRINTLINES_HPP
The resulting "Include Guard" configuration is then:
   #ifndef C1A4E2_PRINTLINES_HPP
   #define C1A4E2_PRINTLINES_HPP
      ...all code goes here...
   #endif
Although comments may be placed outside an "Include Guard", all code must be placed
inside to prevent potential warnings and failures.

??????????
C1A4E2_PrintLines.hpp(...) warning W528:  6 meaningless/cryptic identifiers as follows:
   Line  3, column 21:  a
   Line  3, column 28:  b
   Line  3, column 35:  c
   Line  4, column 21:  a
   Line  4, column 28:  b
   Line  5, column 21:  a
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc.
should convey the meaning but not the value of what they represent.  For example, to
count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no
meaning, whereas names like "studentNumber" or "responseCount" provide some insight
into their purposes.  Although there are some cases in which less meaningful names may
be appropriate, such as when a variable is being used for multiple unrelated purposes
or when names like "x", "y", "z", etc. are used to represent coordinates or abstract
equation variables, this is usually not the situation for exercises in this course.
However, if you believe that your name choice is appropriate in this case, please
contact the instructor to discuss it.

??????????
C1A4E2_PrintLines.hpp(...) advisory A238:  File does not begin with title block.
*** EXPLANATION ***
The first thing in every file must be a title block containing the information
specified in the course document titled "How to Prepare and Submit Assignments".  An
exception is made for header files, in which you may optionally place the title block
just after the first two lines of the always-required "Include Guard" if you prefer.

??????????
C1A4E2_PrintLines.hpp(...) advisory A237:
  The following title block items are either missing or incorrect:
   Student ID
   Email address
   Section ID
   Instructor name
   File name
   Compiler information
*** EXPLANATION ***
Every file must begin with an appropriate title block as illustrated in the course
document titled "How to Prepare and Submit Assignments".  If you have actually provided
```

a Section ID, it is not correct for this course and must instead be 162461. If you have actually provided title block information pertaining to the compiler you are using, please contact the instructor to discuss it.

??????????
C1A4E2_PrintLines-0.cpp(...) warning W647: Unnecessary "<<" operator between string literals
The "<<" operator in the following position is unnecessary:
   Line 18, column 17
*** EXPLANATION ***
String literals separated only by zero or more whitespaces (spaces, tabs, ends of code lines, etc.), are automatically concatenated into a single string by the compiler. Thus, placing the insertion operator, "<<", between them in a cout expression is unnecessary and cluttering. For example, instead of writing:

```
   cout <<
      "ABC\n" <<
      "DEF\n" <<
      "GHI\n";
```

remove all insertion operators that are between string literals, as shown below. In this example the string literals are on separate lines because they represent separate output lines, and that code arrangement provides the best readability.

```
   cout <<
      "ABC\n"
      "DEF\n"
      "GHI\n";
```

??????????
C1A4E2_PrintLines-1.cpp(...) warning W528: 2 meaningless/cryptic identifiers as follows:
   Line 17, column 21: a
   Line 18, column 18: a
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc. should convey the meaning but not the value of what they represent. For example, to count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no meaning, whereas names like "studentNumber" or "responseCount" provide some insight into their purposes. Although there are some cases in which less meaningful names may be appropriate, such as when a variable is being used for multiple unrelated purposes or when names like "x", "y", "z", etc. are used to represent coordinates or abstract equation variables, this is usually not the situation for exercises in this course. However, if you believe that your name choice is appropriate in this case, please contact the instructor to discuss it.

??????????
C1A4E2_PrintLines-2.cpp(...) warning W528: 4 meaningless/cryptic identifiers as follows:
   Line 17, column 21: a
   Line 17, column 28: b
   Line 18, column 38: b
   Line 20, column 22: a
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc. should convey the meaning but not the value of what they represent. For example, to count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no meaning, whereas names like "studentNumber" or "responseCount" provide some insight into their purposes. Although there are some cases in which less meaningful names may be appropriate, such as when a variable is being used for multiple unrelated purposes or when names like "x", "y", "z", etc. are used to represent coordinates or abstract equation variables, this is usually not the situation for exercises in this course. However, if you believe that your name choice is appropriate in this case, please

```
contact the instructor to discuss it.

??????????
C1A4E2_PrintLines-2.cpp(...) warning W301:  The body of the following function contains
insufficiently commented code:
   Line 17, function "PrintLines"
*** EXPLANATION ***
This warning is issued when the code WITHIN THE BODY of a function is not commented
sufficiently.  This requirement cannot be met by comments placed in a file's title
block because these should only describe the file's general contents, nor can it be met
by comments placed just before a function's definition because these should only
describe the function's parameters, return value, and general operation.  Instead,
comments related to the details of the code itself should be placed just before or to
the right of that code as appropriate.  For more details please see the file named
"HowAndWhatToComment.pdf", which is attached to this email.

??????????
C1A4E2_PrintLines-2.cpp(...) advisory A205:  1 missing space as follows:
   Line 18, column  8 (between 'r' and '(')

??????????
C1A4E2_PrintLines-3.cpp(...) warning W528:  6 meaningless/cryptic identifiers as
follows:
   Line 17, column 21:  a
   Line 17, column 28:  b
   Line 17, column 35:  c
   Line 18, column 39:  c
   Line 20, column 42:  b
   Line 22, column 26:  a
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc.
should convey the meaning but not the value of what they represent.  For example, to
count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no
meaning, whereas names like "studentNumber" or "responseCount" provide some insight
into their purposes.  Although there are some cases in which less meaningful names may
be appropriate, such as when a variable is being used for multiple unrelated purposes
or when names like "x", "y", "z", etc. are used to represent coordinates or abstract
equation variables, this is usually not the situation for exercises in this course.
However, if you believe that your name choice is appropriate in this case, please
contact the instructor to discuss it.

??????????
C1A4E2_PrintLines-3.cpp(...) warning W301:  The body of the following function contains
insufficiently commented code:
   Line 17, function "PrintLines"
*** EXPLANATION ***
This warning is issued when the code WITHIN THE BODY of a function is not commented
sufficiently.  This requirement cannot be met by comments placed in a file's title
block because these should only describe the file's general contents, nor can it be met
by comments placed just before a function's definition because these should only
describe the function's parameters, return value, and general operation.  Instead,
comments related to the details of the code itself should be placed just before or to
the right of that code as appropriate.  For more details please see the file named
"HowAndWhatToComment.pdf", which is attached to this email.

??????????
C1A4E2_PrintLines-3.cpp(...) recommendation R151:  "Landscape" page orientation
detected
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
```

Of the 26 total lines in your file, 1 of them (11) exceeds the 80 "portrait" orientation column limit. As a result "landscape" orientation has been selected instead, which allows 110 columns. However, portrait is more common and is usually preferred because shorter lines are more readable. Because none of your code exceeds 80 columns, neither should your comments. Long lines are often caused by unnecessarily long identifier names, which may or may not be a problem in your code. The recommended 80-column limit can be accidentally exceeded if an editor does not provide a clear indication of which columns characters are in, but when configured properly any good code editor will provide it by either displaying column numbers directly, displaying a vertical column "guideline", highlighting characters that go past a specified column, etc. Please consult your editor's documentation to see if it can provide a convenient column number indication to help with your code formatting. If it cannot, I recommend using one that can.

??????????
C1A4E2_PrintLines-3.cpp(...) advisory A205: 1 missing space as follows:
   Line 20, column 12 (between 'r' and '(')

1 80

```
1  #ifndef C1A4E2_PRINTLINES
2  #define C1A4E2_PRINTLINES
3  void PrintLines(int a, int b, int c);
4  void PrintLines(int a, int b);
5  void PrintLines(int a);
6  void PrintLines();
7  #endif
```

1 80

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/06/2022
7  // C1A4E2_main.c
8  // Win10
9  // g++ 11.2.0
10 //
11 // This program prints a specified character in a variety of ways
12 //
13
14 #include <iostream>
15 using namespace std;
16
17 #include "C1A4E2_PrintLines.hpp"
18
19
20 int main() {
21     int numIters = 2;
22     for (int i = 0; i < numIters; i++)
23     {
24         //Get input
25         char input1;
26         int input2, input3;
27         cout << "Input a character and two integers separated by spaces: ";
28         cin >> input1 >> input2 >> input3;
29         //Call the functions
30         PrintLines(int(input1), input2, input3);
31         PrintLines(int(input1), input2);
32         PrintLines(int(input1));
33         PrintLines();
34     }
35 }
```

```
1
//
// Phillip Ward U09339367
// Phillip.Ward@seagate.com
// C/C++ Programming I
// 162461 Ray Mitchell
// 02/06/2022
// C1A4E2_PrintLines-0.cpp
// Win10
// g++ 11.2.0
//
// Contains a function to print a character
//
#include <iostream>

using namespace std;

void PrintLines() {
    cout << "Z" << "\n";
}
```

1 80

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/06/2022
7  // C1A4E2_PrintLines-1.cpp
8  // Win10
9  // g++ 11.2.0
10 //
11 // Contains a function to print a character
12 //
13 #include <iostream>
14
15 using namespace std;
16
17 void PrintLines(int a) {
18     cout << char(a) << "\n";
19 }
```

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/06/2022
7  // C1A4E2_PrintLines-2.cpp
8  // Win10
9  // g++ 11.2.0
10 //
11 // Contains a function to print a character a number of times on one line
12 //
13 #include <iostream>
14
15 using namespace std;
16
17 void PrintLines(int a, int b) {
18     for(int numChars = 0; numChars < b; numChars++)
19     {
20         cout << char(a);
21     }
22     cout << "\n";
23 }
```

1 80 81 <—— Not Recommended ——> 110

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/06/2022
7  // C1A4E2_PrintLines-3.cpp
8  // Win10
9  // g++ 11.2.0
10 //
11 // Contains a function to print a character a number of times on a number of lines
12 //
13 #include <iostream>
14
15 using namespace std;
16
17 void PrintLines(int a, int b, int c) {
18     for (int numLines = 0; numLines < c; numLines++)
19     {
20         for(int numChars = 0; numChars < b; numChars++)
21         {
22             cout << char(a);
23         }
24         cout << "\n";
25     }
26 }
```

```
********** C1 ASSIGNMENT 4 EXERCISE 2 AUTOMATIC PROGRAM RUN RESULTS **********

************* THE RESULTS BELOW HAVE BEEN PARTIALLY CHECKED AND **************
*************   NO ERRORS WERE FOUND.  HOWEVER, THIS DOES NOT   **************
*************  NECESSARILY MEAN THAT THERE ARE NO ERRORS.  THE  **************
*************  INSTRUCTOR WILL DO A MORE THOROUGH CHECK DURING  **************
*************                   MANUAL GRADING.                  **************


------------------------------ START OF 1ST RUN ------------------------------

Input a character and two integers separated by spaces: J 2 3
JJ
JJ
JJ
JJ
J
Z
Input a character and two integers separated by spaces: @ 25 5
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@
Z

------------------------------- END OF 1ST RUN -------------------------------

------------------------------ START OF 2ND RUN ------------------------------

Input a character and two integers separated by spaces: X 6 15
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
X
Z
Input a character and two integers separated by spaces: % 4 0
%%%%
%
Z

------------------------------- END OF 2ND RUN -------------------------------

------------------------------ START OF 3RD RUN ------------------------------
```

```
Input a character and two integers separated by spaces: b 0 4




b
Z
Input a character and two integers separated by spaces: $ 0 0

$
Z

------------------------------ END OF 3RD RUN -------------------------------
```

THIS WAS SENT FROM A NOTIFICATION-ONLY ADDRESS THAT CANNOT ACCEPT INCOMING MAIL.
For help please contact the instructor at the email address provided on the "Home" page of the course's Canvas website. The assignment checker DOES NOT GRADE your submissions but merely reports on issues so you can avoid credit loss by making corrections and resubmitting. ALL GRADING IS DONE MANUALLY BY THE INSTRUCTOR after the assignment deadline based solely upon the NEWEST submission of each exercise that was submitted BEFORE THE ASSIGNMENT DEADLINE. NO CREDIT will be given for anything submitted after the deadline.

From: Phillip Ward <mailto:phillip.ward@seagate.com>
Subject: C1A4E3_162461_U09339367
Submitted: 2/6/2022 2:53:35 PM PST
Course: C/C++ Programming I (Section 162461)
Student's name: Phillip Ward
Contact email: phillip.ward@seagate.com
Student ID: U09339367
Assignment 4, Exercise 3 (001449690M01005X93449)
Exercise point value: 3
Files submitted:
C1A4E3_main.cpp
C1A4E3_PrintLines.cpp
C1A4E3_PrintLines.hpp

"Static analysis" results:
23 warnings as follows:
5 magic number warnings (custom validator);
18 poor practice warnings (custom validator);
3 advisories as follows:
1 inter-token spacing advisory (custom validator);
2 miscellaneous advisories (custom validator);
2 recommendations;
"Runtime" results:
Program ran - No errors detected during preliminary testing (SEE ATTACHMENT);

STANDARD GRADING POLICY:
The MINIMUM deduction is the greater of the following for static analysis issues plus a possible additional deduction for runtime issues, if any:
100% if any "goto" statement is used, else
~45% if any compiler or linker error, else
~25% if any warning, else
~15% if any advisory, else
0% if any recommendation.

C1A4E3: YOUR MINIMUM DEDUCTION: 0.8 points (~25%) To avoid deductions please correct this exercise and resubmit to the assignment checker before the assignment deadline.

The custom validator found 28 problems.
(http://www.MeanOldTeacher.com/AssignmentCheckerKnownIssues.pdf)

??????????
C1A4E3_main.cpp(...) warning W740: Scope of one or more variables is too wide in function "main".
*** EXPLANATION ***
Good practice dictates that whenever practical the scope of variables and other program items should be as small as possible to lessen the chance they will be accessed by parts of the code that should not have access to them. This is a similar philosophy to that used when granting security clearances, that is, the fewer people that have security clearances the less the likelihood of a security breach. In your code the

non-constant variable named numIters has been declared prior to the "for" statement that begins on line 22. If this variable is being used to specify the loop count limit it should be declared as "const". Otherwise it should be declared within the body of the loop because, if your code is written correctly, it will only be used there and will not need to retain its value between loop body executions. For more details please see the file named "LimitingTheScopeOfVariables.pdf", which is attached to this email.

??????????
C1A4E3_main.cpp(26) warning W357: Inappropriate "Magic Number" in identifier "input2" on line 26, column 13: 2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment but can also refer to an embedded character literal or string literal, or to an identifier named for the value it represents rather than its purpose. For more details please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to this email.

??????????
C1A4E3_main.cpp(28) warning W357: Inappropriate "Magic Number" in identifier "input2" on line 28, column 26: 2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment but can also refer to an embedded character literal or string literal, or to an identifier named for the value it represents rather than its purpose. For more details please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to this email.

??????????
C1A4E3_main.cpp(30) warning W357: Inappropriate "Magic Number" in identifier "input2" on line 30, column 33: 2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment but can also refer to an embedded character literal or string literal, or to an identifier named for the value it represents rather than its purpose. For more details please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to this email.

??????????
C1A4E3_main.cpp(31) warning W357: Inappropriate "Magic Number" in identifier "input2" on line 31, column 33: 2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment but can also refer to an embedded character literal or string literal, or to an identifier named for the value it represents rather than its purpose. For more details please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to this email.

??????????
C1A4E3_main.cpp(35) recommendation R150: No return statement in function "main".
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
The language standards permit the "main" function to be implemented without a return statement, in which case a value of 0 will be automatically returned if the end of the function's closing brace is reached. However, the most accepted programming practice is to provide an explicit return statement anyway just for consistency with other functions, all of which require an explicit return statement if they are not declared to return void.

??????????

```
C1A4E3_main.cpp(...) warning W528:  3 meaningless/cryptic identifiers as follows:
   Line 22, column 14:  i
   Line 22, column 21:  i
   Line 22, column 35:  i
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc.
should convey the meaning but not the value of what they represent.  For example, to
count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no
meaning, whereas names like "studentNumber" or "responseCount" provide some insight
into their purposes.  Although there are some cases in which less meaningful names may
be appropriate, such as when a variable is being used for multiple unrelated purposes
or when names like "x", "y", "z", etc. are used to represent coordinates or abstract
equation variables, this is usually not the situation for exercises in this course.
However, if you believe that your name choice is appropriate in this case, please
contact the instructor to discuss it.

??????????
C1A4E3_main.cpp(...) warning W901:  Unnecessary casts as follows:
   Line 30, column 20:  int()
   Line 31, column 20:  int()
   Line 32, column 20:  int()
*** EXPLANATION ***
A cast is typically only necessary for the reasons listed below, but neither should be
present in the code for this particular exercise unless it has been implemented in an
unnecessarily complex manner:
1. An implicit type conversion that causes a warning, OR
2. A desired type conversion that does not occur implicitly.

??????????
C1A4E3_main.cpp(...) warning W350:  1 inappropriate "Magic Number" as follows:
   Line 21, column 20:  2
*** EXPLANATION ***
The term "magic number" most commonly refers to a number embedded in code or comment
but can also refer to an embedded character literal or string literal, or to an
identifier named for the value it represents rather than its purpose.  For more details
please see the file named "AvoidingInappropriateMagicNumbers.pdf", which is attached to
this email.  NOTE: If you are getting this warning about literal values in an array
initializer list and using indentifiers to represent them is not appropriate, try
declaring the array constant.  If this does not fix the problem or if doing so causes a
compiler error, please contact the instructor to discuss it.

??????????
C1A4E3_main.cpp(...) advisory A237:
  The following title block item is either missing or incorrect:
   File name
*** EXPLANATION ***
Every file must begin with an appropriate title block as illustrated in the course
document titled "How to Prepare and Submit Assignments".

??????????
C1A4E3_main.cpp(...) advisory A206:  Unwanted blank line as follows:
   Line  18
*** EXPLANATION ***
Although thoughtfully placed blank lines can make code more readable, excessive or
inappropriately placed blank lines only reduce readablity.

??????????
C1A4E3_PrintLines.cpp(...) warning W528:  6 meaningless/cryptic identifiers as follows:
   Line 17, column 21:  a
   Line 17, column 28:  b
```

```
   Line 17, column 35:  c
   Line 18, column 39:  c
   Line 20, column 42:  b
   Line 22, column 26:  a
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc.
should convey the meaning but not the value of what they represent.  For example, to
count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no
meaning, whereas names like "studentNumber" or "responseCount" provide some insight
into their purposes.  Although there are some cases in which less meaningful names may
be appropriate, such as when a variable is being used for multiple unrelated purposes
or when names like "x", "y", "z", etc. are used to represent coordinates or abstract
equation variables, this is usually not the situation for exercises in this course.
However, if you believe that your name choice is appropriate in this case, please
contact the instructor to discuss it.

??????????
C1A4E3_PrintLines.cpp(...) warning W301:  The body of the following function contains
insufficiently commented code:
   Line 17, function "PrintLines"
*** EXPLANATION ***
This warning is issued when the code WITHIN THE BODY of a function is not commented
sufficiently.  This requirement cannot be met by comments placed in a file's title
block because these should only describe the file's general contents, nor can it be met
by comments placed just before a function's definition because these should only
describe the function's parameters, return value, and general operation.  Instead,
comments related to the details of the code itself should be placed just before or to
the right of that code as appropriate.  For more details please see the file named
"HowAndWhatToComment.pdf", which is attached to this email.

??????????
C1A4E3_PrintLines.cpp(...) recommendation R151:  "Landscape" page orientation detected
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
Of the 26 total lines in your file, 1 of them (11) exceeds the 80 "portrait"
orientation column limit.  As a result "landscape" orientation has been selected
instead, which allows 110 columns.  However, portrait is more common and is usually
preferred because shorter lines are more readable.  Because none of your code exceeds
80 columns, neither should your comments.  Long lines are often caused by unnecessarily
long identifier names, which may or may not be a problem in your code.  The recommended
80-column limit can be accidentally exceeded if an editor does not provide a clear
indication of which columns characters are in, but when configured properly any good
code editor will provide it by either displaying column numbers directly, displaying a
vertical column "guideline", highlighting characters that go past a specified column,
etc.  Please consult your editor's documentation to see if it can provide a convenient
column number indication to help with your code formatting.  If it cannot, I recommend
using one that can.

??????????
C1A4E3_PrintLines.cpp(...) advisory A205:  1 missing space as follows:
   Line 20, column 12 (between 'r' and '(')

??????????
C1A4E3_PrintLines.hpp(14) warning W551:  Improper "Include Guard" naming:
C1A4E3_PRINTLINES
*** EXPLANATION ***
An "Include Guard" (appendix note D.2) must be implemented in every header file (but
never in an implementation file) to prevent multiple inclusions of its contents if it
is included more than once.  The most standard "Include Guard" naming convention is the
uppercase name of the file itself with all non-alphanumeric characters replaced by
```

underscores and, if the name starts with a number, an underscore prepended. For your file:
   C1A4E3_PrintLines.hpp
the "Include Guard" name must be (or at least must end with):
   C1A4E3_PRINTLINES_HPP
The resulting "Include Guard" configuration is then:
```
   #ifndef C1A4E3_PRINTLINES_HPP
   #define C1A4E3_PRINTLINES_HPP
      ...all code goes here...
   #endif
```
Although comments may be placed outside an "Include Guard", all code must be placed inside to prevent potential warnings and failures.

??????????
C1A4E3_PrintLines.hpp(...) warning W528: 3 meaningless/cryptic identifiers as follows:
   Line 15, column 21: a
   Line 15, column 34: b
   Line 15, column 45: c
*** EXPLANATION ***
Whenever possible and practical the names used for variables, functions, macros, etc. should convey the meaning but not the value of what they represent. For example, to count executions of a loop's body names like "i", "j", "k", etc. convey absolutely no meaning, whereas names like "studentNumber" or "responseCount" provide some insight into their purposes. Although there are some cases in which less meaningful names may be appropriate, such as when a variable is being used for multiple unrelated purposes or when names like "x", "y", "z", etc. are used to represent coordinates or abstract equation variables, this is usually not the situation for exercises in this course. However, if you believe that your name choice is appropriate in this case, please contact the instructor to discuss it.

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/06/2022
7  // C1A4E3_PrintLines.hpp
8  // Win10
9  // g++ 11.2.0
10 //
11 // Header file with PrintLine function declaration
12 //
13 #ifndef C1A4E3_PRINTLINES
14 #define C1A4E3_PRINTLINES
15 void PrintLines(int a = 'Z', int b = 1, int c = 1);
16 #endif
```

```
1
//
// Phillip Ward U09339367
// Phillip.Ward@seagate.com
// C/C++ Programming I
// 162461 Ray Mitchell
// 02/06/2022
// C1A4E3_main.c
// Win10
// g++ 11.2.0
//
// This program prints a specified character in a variety of ways
//

#include <iostream>
using namespace std;

#include "C1A4E3_PrintLines.hpp"


int main() {
    int numIters = 2;
    for (int i = 0; i < numIters; i++)
    {
        //Get input
        char input1;
        int input2, input3;
        cout << "Input a character and two integers separated by spaces: ";
        cin >> input1 >> input2 >> input3;
        //Call the functions
        PrintLines(int(input1), input2, input3);
        PrintLines(int(input1), input2);
        PrintLines(int(input1));
        PrintLines();
    }
}
```

1 80 81 <—— Not Recommended ——> 110

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/06/2022
7  // C1A4E3_PrintLines.cpp
8  // Win10
9  // g++ 11.2.0
10 //
11 // Contains a function to print a character a number of times on a number of lines
12 //
13 #include <iostream>
14
15 using namespace std;
16
17 void PrintLines(int a, int b, int c) {
18     for (int numLines = 0; numLines < c; numLines++)
19     {
20         for(int numChars = 0; numChars < b; numChars++)
21         {
22             cout << char(a);
23         }
24         cout << "\n";
25     }
26 }
```

```
********** C1 ASSIGNMENT 4 EXERCISE 3 AUTOMATIC PROGRAM RUN RESULTS **********

************* THE RESULTS BELOW HAVE BEEN PARTIALLY CHECKED AND **************
*************   NO ERRORS WERE FOUND.  HOWEVER, THIS DOES NOT   **************
*************  NECESSARILY MEAN THAT THERE ARE NO ERRORS.  THE  **************
*************  INSTRUCTOR WILL DO A MORE THOROUGH CHECK DURING  **************
*************                   MANUAL GRADING.                  **************


------------------------------ START OF 1ST RUN ------------------------------

Input a character and two integers separated by spaces: J 2 3
JJ
JJ
JJ
JJ
J
Z
Input a character and two integers separated by spaces: @ 25 5
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@
@
Z

------------------------------ END OF 1ST RUN -------------------------------

------------------------------ START OF 2ND RUN ------------------------------

Input a character and two integers separated by spaces: X 6 15
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
XXXXXX
X
Z
Input a character and two integers separated by spaces: % 4 0
%%%%
%
Z

------------------------------ END OF 2ND RUN -------------------------------

------------------------------ START OF 3RD RUN ------------------------------
```

```
Input a character and two integers separated by spaces: b 0 4




b
Z
Input a character and two integers separated by spaces: $ 0 0

$
Z

------------------------------ END OF 3RD RUN -------------------------------
```

THIS WAS SENT FROM A NOTIFICATION-ONLY ADDRESS THAT CANNOT ACCEPT INCOMING MAIL.
For help please contact the instructor at the email address provided on the "Home" page of the course's Canvas website. The assignment checker DOES NOT GRADE your submissions but merely reports on issues so you can avoid credit loss by making corrections and resubmitting. ALL GRADING IS DONE MANUALLY BY THE INSTRUCTOR after the assignment deadline based solely upon the NEWEST submission of each exercise that was submitted BEFORE THE ASSIGNMENT DEADLINE. NO CREDIT will be given for anything submitted after the deadline.

From: Phillip Ward <mailto:phillip.ward@seagate.com>
Subject: C1A4E4_162461_U09339367
Submitted: 2/6/2022 3:28:26 PM PST
Course: C/C++ Programming I (Section 162461)
Student's name: Phillip Ward
Contact email: phillip.ward@seagate.com
Student ID: U09339367
Assignment 4, Exercise 4 (001578073M01005X60578)
Exercise point value: 4
Files submitted:
C1A4E4_main.cpp
C1A4E4_MaxOf.h

"Static analysis" results:
5 warnings as follows:
3 poor practice warnings (custom validator);
2 unsafe practice warnings (custom validator);
5 advisories as follows:
3 inter-token spacing advisories (custom validator);
2 miscellaneous advisories (custom validator);
2 recommendations;
"Runtime" results:
Program ran - No errors detected during preliminary testing (SEE ATTACHMENT);

STANDARD GRADING POLICY:
The MINIMUM deduction is the greater of the following for static analysis issues plus a possible additional deduction for runtime issues, if any:
100% if any "goto" statement is used, else
~45% if any compiler or linker error, else
~25% if any warning, else
~15% if any advisory, else
0% if any recommendation.

C1A4E4: YOUR MINIMUM DEDUCTION: 1 point (~25%) To avoid deductions please correct this exercise and resubmit to the assignment checker before the assignment deadline.

The custom validator found 12 problems.
(http://www.MeanOldTeacher.com/AssignmentCheckerKnownIssues.pdf)

??????????
C1A4E4_main.cpp(29) recommendation R150: No return statement in function "main".
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
The language standards permit the "main" function to be implemented without a return statement, in which case a value of 0 will be automatically returned if the end of the function's closing brace is reached. However, the most accepted programming practice is to provide an explicit return statement anyway just for consistency with other functions, all of which require an explicit return statement if they are not declared to return void.

```
??????????
C1A4E4_main.cpp(...) warning W647:  Unnecessary "<<" operator between string literals
The "<<" operators in the following positions are unnecessary:
   Line 26, column 10
   Line 28, column 10
*** EXPLANATION ***
String literals separated only by zero or more whitespaces (spaces, tabs, ends of code
lines, etc.), are automatically concatenated into a single string by the compiler.
Thus, placing the insertion operator, "<<", between them in a cout expression is
unnecessary and cluttering.  For example, instead of writing:
   cout <<
      "ABC\n" <<
      "DEF\n" <<
      "GHI\n";
remove all insertion operators that are between string literals, as shown below.  In
this example the string literals are on separate lines because they represent separate
output lines, and that code arrangement provides the best readability.
   cout <<
      "ABC\n"
      "DEF\n"
      "GHI\n";

??????????
C1A4E4_main.cpp(...) advisory A206:  Unwanted blank line as follows:
   Line  18
*** EXPLANATION ***
Although thoughtfully placed blank lines can make code more readable, excessive or
inappropriately placed blank lines only reduce readablity.

??????????
C1A4E4_MaxOf.h(14) warning W551:  Improper "Include Guard" naming:  C1A4E4_MAXOF
*** EXPLANATION ***
An "Include Guard" (appendix note D.2) must be implemented in every header file (but
never in an implementation file) to prevent multiple inclusions of its contents if it
is included more than once.  The most standard "Include Guard" naming convention is the
uppercase name of the file itself with all non-alphanumeric characters replaced by
underscores and, if the name starts with a number, an underscore prepended.  For your
file:
   C1A4E4_MaxOf.h
the "Include Guard" name must be (or at least must end with):
   C1A4E4_MAXOF_H
The resulting "Include Guard" configuration is then:
   #ifndef C1A4E4_MAXOF_H
   #define C1A4E4_MAXOF_H
      ...all code goes here...
   #endif
Although comments may be placed outside an "Include Guard", all code must be placed
inside to prevent potential warnings and failures.

??????????
C1A4E4_MaxOf.h(...) warning W559:  Extraneous semicolons as follows:
   Line 17, column 100
   Line 19, column 49
*** EXPLANATION ***
In many contexts a semicolon is used to terminate a statement, while in others it acts
as a complete statement in itself (known as a "null" statement).  Although "null"
statements are sometimes appropriate, placing one after the closing brace of a "block"
statement as you have done serves no purpose and can cause problems in some contexts.
```

```
??????????
C1A4E4_MaxOf.h(...) recommendation R151:  "Landscape" page orientation detected
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
Of the 20 total lines in your file, 2 of them (17 and 18) exceed the 80 "portrait"
orientation column limit.  As a result "landscape" orientation has been selected
instead, which allows 110 columns.  However, portrait is more common and is usually
preferred because shorter lines are more readable.  Long lines are often caused by
unnecessarily long identifier names, which may or may not be a problem in your code.
The recommended 80-column limit can be accidentally exceeded if an editor does not
provide a clear indication of which columns characters are in, but when configured
properly any good code editor will provide it by either displaying column numbers
directly, displaying a vertical column "guideline", highlighting characters that go
past a specified column, etc.  Please consult your editor's documentation to see if it
can provide a convenient column number indication to help with your code formatting.
If it cannot, I recommend using one that can.

??????????
C1A4E4_MaxOf.h(19) advisory A228:  Improper brace placement:
return(fMaxOf2(fMaxOf2(num1, num2), num3));};
*** EXPLANATION ***
Do not place a closing brace after code on the same line if the corresponding opening
brace is on another line.

??????????
C1A4E4_MaxOf.h(...) advisory A205:  2 missing spaces as follows:
   Line 17, column 63 (between ')' and '{')
   Line 18, column 81 (between ')' and '{')

??????????
C1A4E4_MaxOf.h(...) advisory A204:  1 unwanted space as follows:
   Line 17, column 70 (between 'n' and '(')

??????????
C1A4E4_MaxOf.h(...) warning W211:  The parentheses indicated by the ^ markers are
missing from your code but are required:
*** NOTE *** If you are viewing this as email text and the markers are misaligned,
change your viewer's font to "Courier New" or a similar "monospace" font.
   16. #define mMaxOf3(num1, num2, num3) (mMaxOf2(mMaxOf2((num1), (num2)), (num3)))
                                          ^                ^    ^  ^    ^   ^    ^ ^
*** EXPLANATION ***
Always parenthesize:
   1. A multi-token macro replacement list, and
   2. Every usage of a macro's parameters in its replacement list.
Do not parenthesize:
   3. A single-token replacement list that is not a parameter of that macro, or
   4. Anything else in the replacement list unless required by language syntax.
```

1 80 81 <—— Not Recommended ——> 110

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/06/2022
7  // C1A4E4_MaxOf.h
8  // Win10
9  // g++ 11.2.0
10 //
11 // Header file function and macro definitions
12 //
13 #ifndef C1A4E4_MAXOF
14 #define C1A4E4_MAXOF
15 #define mMaxOf2(num1, num2) ((num1) > (num2) ? (num1) : (num2))
16 #define mMaxOf3(num1, num2, num3) mMaxOf2(mMaxOf2(num1, num2), num3)
17 inline long double fMaxOf2(long double num1, long double num2){return (num1 > num2 ? num1 : num2);};
18 inline long double fMaxOf3(long double num1, long double num2, long double num3){
19     return(fMaxOf2(fMaxOf2(num1, num2), num3));};
20 #endif
```

```
1
//
// Phillip Ward U09339367
// Phillip.Ward@seagate.com
// C/C++ Programming I
// 162461 Ray Mitchell
// 02/06/2022
// C1A4E4_main.cpp
// Win10
// g++ 11.2.0
//
// This program prints a specified character in a variety of ways
//

#include <iostream>
using namespace std;

#include "C1A4E4_MaxOf.h"


int main() {
    long double input1, input2, input3;
    cout << "Input three decimal numbers separated by spaces: ";
    cin >> input1 >> input2 >> input3;
    //Call the functions and print the results
    cout << "mMaxOf3(" << input1 << ", " << input2 << ", " << input3 << ") "
         << "returned " <<  mMaxOf3(input1, input2, input3) << "\n";
    cout << "fMaxOf3(" << input1 << ", " << input2 << ", " << input3 << ") "
         << "returned " <<  fMaxOf3(input1, input2, input3) << "\n";
}
```

```
********** C1 ASSIGNMENT 4 EXERCISE 4 AUTOMATIC PROGRAM RUN RESULTS **********

************ THE RESULTS BELOW HAVE BEEN PARTIALLY CHECKED AND **************
************   NO ERRORS WERE FOUND.  HOWEVER, THIS DOES NOT   **************
************  NECESSARILY MEAN THAT THERE ARE NO ERRORS.  THE  **************
************  INSTRUCTOR WILL DO A MORE THOROUGH CHECK DURING  **************
************                   MANUAL GRADING.                    **************


------------------------------ START OF 1ST RUN ------------------------------

Input three decimal numbers separated by spaces: 3 3 3
mMaxOf3(3, 3, 3) returned 3
fMaxOf3(3, 3, 3) returned 3


------------------------------- END OF 1ST RUN -------------------------------

------------------------------ START OF 2ND RUN ------------------------------

Input three decimal numbers separated by spaces: -7.98 7.13 -18.2
mMaxOf3(-7.98, 7.13, -18.2) returned 7.13
fMaxOf3(-7.98, 7.13, -18.2) returned 7.13


------------------------------- END OF 2ND RUN -------------------------------

------------------------------ START OF 3RD RUN ------------------------------

Input three decimal numbers separated by spaces: 2000.45 0 2000.46
mMaxOf3(2000.45, 0, 2000.46) returned 2000.46
fMaxOf3(2000.45, 0, 2000.46) returned 2000.46


------------------------------- END OF 3RD RUN -------------------------------

------------------------------ START OF 4TH RUN ------------------------------

Input three decimal numbers separated by spaces: 33.3 54e-2 86e-1
mMaxOf3(33.3, 0.54, 8.6) returned 33.3
fMaxOf3(33.3, 0.54, 8.6) returned 33.3


------------------------------- END OF 4TH RUN -------------------------------

------------------------------ START OF 5TH RUN ------------------------------

Input three decimal numbers separated by spaces: 86e-1 86e-1 54e-2
mMaxOf3(8.6, 8.6, 0.54) returned 8.6
fMaxOf3(8.6, 8.6, 0.54) returned 8.6


------------------------------- END OF 5TH RUN -------------------------------

------------------------------ START OF 6TH RUN ------------------------------

Input three decimal numbers separated by spaces: -0 0 0
mMaxOf3(-0, 0, 0) returned 0
fMaxOf3(-0, 0, 0) returned 0
```

---------- END OF 6TH RUN ----------