Consolidated Assignment 7 Report

This report contains the graded results for the newest of each exercise submitted to the assignment checker prior to 3/2/2022 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: 2/27/2022 6:31:44 PM PST
   Exercise 1: 2/27/2022 1:42:28 PM PST
   Exercise 2: 2/27/2022 7:19:50 PM PST

Credit to be deducted for uncorrected assignment checker issue(s):
   Exercise 2: 1.8 points (~25%) minimum plus a runtime issue
                deduction to be determined.

Score (out of 20 possible): __11.5__

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: C1A7E0_162461_U09339367
Submitted: 2/27/2022 6:31:44 PM PST
Course: C/C++ Programming I (Section 162461)
Student's name: Phillip Ward
Contact email: phillip.ward@seagate.com
Student ID: U09339367
Assignment 7, Exercise 0 (001706107M01005X68706)
Exercise point value: 6
File submitted:
C1A7E0_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.

-4

Phillip Ward U09339367
Phillip.Ward@seagate.com
C/C++ Programming I
162461 Ray Mitchell
02/27/2022
C1A7E0_Quiz.txt
Quiz Answers

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

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: C1A7E1_162461_U09339367
Submitted: 2/27/2022 1:42:28 PM PST
Course: C/C++ Programming I (Section 162461)
Student's name: Phillip Ward
Contact email: phillip.ward@seagate.com
Student ID: U09339367
Assignment 7, Exercise 1 (001546261M01005X17546)
Exercise point value: 7
Files submitted:
C1A7E1_DetermineElapsedTime.cpp
C1A7E1_main.cpp
C1A7E1_MyTime.h

"Static analysis" results:
No "static" issues;
"Runtime" results:
Program ran - No errors detected during preliminary testing (SEE ATTACHMENT);

1 80

```
1  //
2  // Phillip Ward U09339367
3  // Phillip.Ward@seagate.com
4  // C/C++ Programming I
5  // 162461 Ray Mitchell
6  // 02/22/2022
7  // C1A7E1_MyTime.h
8  // Win10
9  // g++ 11.2.0
10 //
11 // Header file that contains the MyTime prototype
12 //
13 #ifndef C1A7E1_MYTIME_H
14 #define C1A7E1_MYTIME_H
15 struct MyTime {int hours, minutes, seconds;};
16 #endif
```

```cpp
//
// Phillip Ward U09339367
// Phillip.Ward@seagate.com
// C/C++ Programming I
// 162461 Ray Mitchell
// 02/22/2022
// C1A7E1_main.cpp
// Win10
// g++ 11.2.0
//
// This file contains a program to get the time elapsed between input times
//
#include "C1A7E1_MyTime.h"
#include <iostream>
#include <iomanip>
using namespace std;
static const int NUM_RUNS = 3;
MyTime *DetermineElapsedTime(const MyTime *startTime, const MyTime *endTime);
int main()
{
    //set the formatting for the output
    cout << setfill('0');
    for (int numRuns = NUM_RUNS; numRuns > 0; numRuns--)
    {
        MyTime start, end, *difference;
        char delim;
        //Get input
        cout << "Enter start time (HH:MM:SS): ";
        cin >> start.hours >> delim >> start.minutes >> delim >> start.seconds;
        cout << "Enter end time (HH:MM:SS): ";
        cin >> end.hours >> delim >> end.minutes >> delim >> end.seconds;
        //Get the elapsed time
        difference = DetermineElapsedTime(&start, &end);
        //Print results
        cout << "The time elapsed from " << setw(2) << start.hours << delim
             << setw(2) << start.minutes << delim << setw(2) << start.seconds
             << " to " << setw(2) << end.hours << delim << setw(2)
             << end.minutes << delim << setw(2) << end.seconds << " is " <<
             setw(2) << difference->hours << delim
             << setw(2) << difference->minutes << delim << setw(2) <<
             difference->seconds << "\n";
    }
    return(0);
}
```

```
1   //
2   // Phillip Ward U09339367
3   // Phillip.Ward@seagate.com
4   // C/C++ Programming I
5   // 162461 Ray Mitchell
6   // 02/22/2022
7   // C1A7E1_DetermineElapsedTime.cpp
8   // Win10
9   // g++ 11.2.0
10  //
11  // File that contains a function to compute
12  // time elapsed between two given times
13  //
14  #include "C1A7E1_MyTime.h"
15  const int HOURS_PER_DAY = 24;
16  const int MINUTES_PER_HOUR = 60;
17  const int SECONDS_PER_MINUTE = 60;
18
19  MyTime *DetermineElapsedTime(const MyTime *startTime, const MyTime *endTime)
20  {
21      static MyTime elapsedTime;
22      //convert times to seconds
23      long startTimeInSecs = startTime->hours * MINUTES_PER_HOUR
24          * SECONDS_PER_MINUTE + startTime->minutes * SECONDS_PER_MINUTE
25          + startTime->seconds;
26      long endTimeInSecs = endTime->hours * MINUTES_PER_HOUR * SECONDS_PER_MINUTE
27          + endTime->minutes * SECONDS_PER_MINUTE + endTime->seconds;
28      //check to see if the end time should be interpreted as tomorrow
29      if (endTimeInSecs <= startTimeInSecs)
30      {
31          endTimeInSecs += HOURS_PER_DAY * MINUTES_PER_HOUR * SECONDS_PER_MINUTE;
32      }
33      //find the elapsed time
34      long elapsedTimeInSecs = endTimeInSecs - startTimeInSecs;
35      //convert back into hours minutes and seconds
36      elapsedTime.hours = int((elapsedTimeInSecs / SECONDS_PER_MINUTE)
37                                                  / MINUTES_PER_HOUR);
38      elapsedTimeInSecs -= elapsedTime.hours * MINUTES_PER_HOUR
39                                                 * SECONDS_PER_MINUTE;
40      elapsedTime.minutes = int(elapsedTimeInSecs / SECONDS_PER_MINUTE);
41      elapsedTimeInSecs -= elapsedTime.minutes * SECONDS_PER_MINUTE;
42      elapsedTime.seconds = int(elapsedTimeInSecs);
43
44      return(&elapsedTime);
45  }
```

Comment will easily fit onto one line. Why make it harder to read by prematurely splitting it onto more lines than necessary?

Overflow if int is 16 bits wide. Max 16-bit int value is 32767 but 24 * 3600 is 86400.

-0.2

Possible overflow if int is 16 bits wide. Max 16-bit int value is 32,767 but max possible seconds in 23 hours is 23 * 3600 is 82,800.

-0.3

```
********** C1 ASSIGNMENT 7 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 ------------------------------

Enter start time (HH:MM:SS): 00:00:00 00:00:00
Enter end time (HH:MM:SS): The time elapsed from 00:00:00 to 00:00:00 is 24:00:00
Enter start time (HH:MM:SS): 12:12:12 13:12:11
Enter end time (HH:MM:SS): The time elapsed from 12:12:12 to 13:12:11 is 00:59:59
Enter start time (HH:MM:SS): 13:12:11 12:12:12
Enter end time (HH:MM:SS): The time elapsed from 13:12:11 to 12:12:12 is 23:00:01

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

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

Enter start time (HH:MM:SS): 00:00:00 23:59:58
Enter end time (HH:MM:SS): The time elapsed from 00:00:00 to 23:59:58 is 23:59:58
Enter start time (HH:MM:SS): 23:59:58 00:00:00
Enter end time (HH:MM:SS): The time elapsed from 23:59:58 to 00:00:00 is 00:00:02
Enter start time (HH:MM:SS): 22:03:27 18:18:49
Enter end time (HH:MM:SS): The time elapsed from 22:03:27 to 18:18:49 is 20:15:22

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

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

Enter start time (HH:MM:SS): 12:23:34 12:23:35
Enter end time (HH:MM:SS): The time elapsed from 12:23:34 to 12:23:35 is 00:00:01
Enter start time (HH:MM:SS): 10:00:59 00:18:22
Enter end time (HH:MM:SS): The time elapsed from 10:00:59 to 00:18:22 is 14:17:23
Enter start time (HH:MM:SS): 09:09:09 02:23:35
Enter end time (HH:MM:SS): The time elapsed from 09:09:09 to 02:23:35 is 17:14:26

------------------------------- 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: C1A7E2_162461_U09339367
Submitted: 2/27/2022 7:19:50 PM PST
Course: C/C++ Programming I (Section 162461)
Student's name: Phillip Ward
Contact email: phillip.ward@seagate.com
Student ID: U09339367
Assignment 7, Exercise 2 (001892317M01005X94892)
Exercise point value: 7
File submitted:
C1A7E2_main.c

-4

"Static analysis" results:
15 warnings as follows:
3 compiler warnings (Clang compiler);
2 compiler warnings (Microsoft compiler);
4 exercise-specific warnings (custom validator);
4 poor practice warnings (custom validator);
1 unsafe practice warning (custom validator);
1 miscellaneous warning (custom validator);
7 advisories as follows:
2 indentation advisories (custom validator);
4 inter-token spacing advisories (custom validator);
1 miscellaneous advisory (custom validator);
3 recommendations;
"Runtime" results:
Program ran - ERRORS WERE DETECTED (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.

C1A7E2: YOUR MINIMUM DEDUCTION: 1.8 points (~25%) plus a runtime issue deduction to be determined. To avoid deductions please correct this exercise and resubmit to the assignment checker before the assignment deadline.

The Clang compiler found 3 problems.
(http://clang.llvm.org/)

??????????
C1A7E2_main.c:36:27: warning: format specifies type 'char *' but the argument has type 'char (*)[100]'
scanf("%s %d %d", &foodName, &lunches[lunchNum].weight,
~~ ^~~~~~~~

```
??????????
C1A7E2_main.c:29:9: warning: unused variable 'weightOz'
    int weightOz, numCalories;
        ^
*** EXPLANATION ***
Variable "weightOz" was not used and should be removed if not needed.  Simply assigning
to, incrementing, decrementing, or changing the value of a variable or other object
does not constitute "use".  This is because if these are the only operations being
performed on it, not doing them at all will not affect the way the program works.

??????????
C1A7E2_main.c:29:19: warning: unused variable 'numCalories'
    int weightOz, numCalories;
                  ^
*** EXPLANATION ***
Variable "numCalories" was not used and should be removed if not needed.  Simply
assigning to, incrementing, decrementing, or changing the value of a variable or other
object does not constitute "use".  This is because if these are the only operations
being performed on it, not doing them at all will not affect the way the program works.



##### The Microsoft compiler found 2 problems. #####
(https://www.visualstudio.com/)

??????????
C1A7E2_main.c(29): warning C4101: 'weightOz': unreferenced local variable

??????????
C1A7E2_main.c(29): warning C4101: 'numCalories': unreferenced local variable



##### The custom validator found 20 problems. #####
(http://www.MeanOldTeacher.com/AssignmentCheckerKnownIssues.pdf)

??????????
C1A7E2_main.c(...) warning W527:  Too many calls to the "strlen" function
*** EXPLANATION ***
You have called the "strlen" function at least 2 times in your code but only once is
necessary for this exercise.  No function should ever be called more than once if the
results will not differ from those of a previous call.  Multiple calls to a function
that returns the length of the same string waste resources and must be avoided.  If
such calls occur within a loop the inefficiency is even worse.  If your code actually
needs to know the length of the same string more than once simply store the result of
the first call to "strlen" in a variable and use that variable in place of the
subsequent "strlen" calls.  If you believe you are getting this warning erroneously,
please contact the instructor to discuss it.

??????????
C1A7E2_main.c(...) warning W572:  No call to the "free" function
*** EXPLANATION ***
In this exercise you must use the standard library "free" function to explicitly free
all dynamically allocated memory after you are finished using it.  You have not done
this.

??????????
C1A7E2_main.c(...) warning W745:  Improper error handling
*** EXPLANATION ***
```

If dynamic memory allocation fails the most common course of action is to display an error message to stderr and terminate the program with an error code. Both of these are required in this exercise but you have not terminated with an error code.

??????????
C1A7E2_main.c(36) warning W525: Missing field width specifier in a scanf %s conversion specification.
*** EXPLANATION ***
To prevent potential target buffer overflow a field width must be specified for every %s and %[] used in any scanf family function (scanf, fscanf, sscanf, vscanf, vfscanf, or vscanf) unless an asterisk is placed after the %. However, the asterisk will cause the corresponding input to be discarded rather than being assigned to a corresponding argument.

??????????
C1A7E2_main.c(...) warning W596: The array listed below should be declared constant or should not be initialized:
   Line 21, column 10: appleString
*** EXPLANATION ***
Any array that is initialized as part of its declaration and whose contents will not be later altered by the program should be declared as constant. Conversely, if an array will be altered it should not normally be initialized at all if any of the initial values will get overwritten without first being used. There are occasional exceptions, but none in this course.

??????????
C1A7E2_main.c(...) warning W925: Use of sizeof on a char type as follows:
   Line 40, column 55: sizeof(char)
*** EXPLANATION ***
By definition, the number of bytes in any char type (char, signed char, or unsigned char) is always 1. If your sizeof expression is being used in a context where a value of 1 is actually needed (addition, subtraction, shifting, function argument, etc.), use a literal 1 instead. Otherwise, if you have used it in a context where a value of 1 makes no difference (multiplication, division, etc.), omit the sizeof expression entirely.

??????????
C1A7E2_main.c(39, 50) warning W629: Unnecessary similar calls to the "strlen" function: strlen(foodName)
*** EXPLANATION ***
Only one call to the "strlen" function is appropriate in this case. Because there can be a significant amount of processing involved in executing a function the same function should never be called more than once if the result will be the same or if the result of a subsequent call can be easily determined from the result of the first call. Instead, the result of the first call should be stored in a variable and that variable used/reused as needed rather than calling the function again.

??????????
C1A7E2_main.c(...) warning W687: 3 overscoped variables as follows:
   Line 28, column 10: foodName
   Line 29, column 9: weightOz
   Line 29, column 19: numCalories
*** EXPLANATION ***
The scope of an identifier (a name) is defined as the portion of code over which it is accessible. The scope of a variable declared inside a block extends from that declaration to the end of that block, where a "block" is commonly defined as a "curly-brace enclosed sequence of 0 or more statements". Good programming practice dictates that the scopes of non-const variables be as small as possible to prevent their values from being changed by code that should not change them. However, because the values of const variables cannot be changed, if being used in place of macros they should be

declared in the same place the macros would have been defined. Otherwise they should be declared first in the function or block that uses them. For more details please see the file named "LimitingTheScopeOfVariables.pdf", which is attached to this email.

??????????
C1A7E2_main.c(...) recommendation R151: "Landscape" page orientation detected
NOTE: There will be no deduction for this but correction is recommended.
*** EXPLANATION ***
Of the 61 total lines in your file, 1 of them (56) 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. 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.

??????????
C1A7E2_main.c(36) advisory A230: Unnecessary split of line 36 onto line 37
*** EXPLANATION ***
Code is usually most readable if certain types of constructs are kept on the same line. Among these are macro definitions, function declarations, function/macro calls, statements using insertion or extraction operators (ie, cout, cin, etc.), the control portions of "if", "switch", "for", "while", and "do" statements, and anything else in parentheses. The line you have split, as indicated above, would be more readable without the split.

??????????
C1A7E2_main.c(...) advisory A202: Incorrectly indented code
Indentation advisories are based upon the expectation that the sizes of all code indents in a file will be appropriate integral multiples of the size of the first code indent, which is 4 columns in this file.
*** 2 incorrect indents as follows:
   Line 25: Your indent = 4; Correct indent = 8
   Line 26: Your indent = 4; Correct indent = 8

??????????
C1A7E2_main.c(...) advisory A205: 3 missing spaces as follows:
   Line 39, column 39 (between 'r' and '*')
   Line 50, column 71 (between '+' and '1')
   Line 50, column 70 (between ')' and '+')

??????????
C1A7E2_main.c(...) advisory A204: 1 unwanted space as follows:
   Line 45, column 21 (between '(' and 's')

??????????
C1A7E2_main.c(...) recommendation R264: Potentially problematic use of C-style comments
NOTE: There will be no deduction for the following but correction is recommended:
   Line 25 column 17 to line 25 column 46
   Line 26 column 27 to line 26 column 74
*** EXPLANATION ***
Although C and C++ style comments (/**/ and // respectively) are demonstrated in the course book, C-style comments are usually undesirable in both C and C++ code for two reasons:

1. C-style comments take more space.
2. Blocks of code containing C-style comments can be more difficult to "comment out" than those containing C++ style comments. Consider the four lines of code below, each of which ends with a C-style comment. Programmers often comment out code blocks during development or debugging by putting a /* at the beginning of the unwanted code and a */ at the end, as shown for lines 2 and 3. However, because it is not legal to have a C-style comment within a C-style comment, a compiler error will occur. If the original comments had instead been C++ style, there would not be a problem.

Example:

```
   Line 1 of code  /* comment 1 */
/*
   Line 2 of code  /* comment 2 */
   Line 3 of code  /* comment 3 */
*/
   Line 4 of code  /* comment 4 */
```

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  // 02/27/2022
7  // C1A7E2_main.c
8  // Win10
9  // g++ 11.2.0
10 //
11 // A program that gets and displays lunch food items
12 //
13 #include <stdio.h>
14 #include <string.h>
15 #include <stdlib.h>
16 #define LUNCH_QTY 3
17 #define NUM_LUNCH_INIT 2
18 #define BUF_SIZE 100
19
20 int main(void) {
21     char appleString[] = "Apple", saladString[] = "Salad";
22
23     struct Food
24     {
25     char *name; /* "name" attribute of food */
26     int weight, calories; /* "weight" and "calories" attributes of food */
27     } lunches[LUNCH_QTY] = {{appleString, 4, 100}, {saladString, 2, 80}};
28     char foodName[BUF_SIZE];
29     int weightOz, numCalories;
30
31     for (int lunchNum = NUM_LUNCH_INIT; lunchNum < LUNCH_QTY; lunchNum++)
32     {
33         //Get input
34         printf("Input the space separated name, weight, "
35                "and caloric content of a food item: ");
36         scanf("%s %d %d", &foodName, &lunches[lunchNum].weight,
37                                      &lunches[lunchNum].calories);
38         //Dynamically allocate memory using malloc()
39         lunches[lunchNum].name = (char*)malloc(strlen(foodName) + 1
40                                                * sizeof(char));
41         //Check if the memory has been successfully allocated by malloc or not
42         if (lunches[lunchNum].name == NULL)
43         {
44             printf("");
45             fprintf( stderr, "Memory allocation failed.\n");
46         }
47         else
48         {
49             //Memory has been successfuly allocated, copy the string over
50             memcpy(lunches[lunchNum].name, foodName, strlen(foodName)+1);
51         }
52     }
53     //Print the lunch table
54     for (int lunchNum = 0; lunchNum < LUNCH_QTY; lunchNum++)
55     {
56         printf("%20s %5d %5d\n", lunches[lunchNum].name, lunches[lunchNum].weight,
57                  lunches[lunchNum].calories);
58     }
59
60     return(0);
61 }
```

```
********** C1 ASSIGNMENT 7 EXERCISE 2 AUTOMATIC PROGRAM RUN RESULTS **********

**************************** ERROR(S) DETECTED ****************************


---------------------------- PURPOSE OF 1ST RUN ----------------------------
Verify a table of food properties.
------------------------- CODE CHANGES FOR 1ST RUN -------------------------
LUNCH_QTY = 6
----------------------------- START OF 1ST RUN -----------------------------
THIS RUN FAILED BECAUSE:
   1. Items in the table are not properly aligned.
   2. A dynamic memory problem was encountered.


Input the space separated name, weight, and caloric content of a food item: blueberries
3 76
Input the space separated name, weight, and caloric content of a food item: sludge 1000
2000
Input the space separated name, weight, and caloric content of a food item: pho 28 302
Input the space separated name, weight, and caloric content of a food item: steak 6 275
               Apple     4   100
               Salad     2    80
         blueberries     3    76
              sludge  1000  2000
                 pho    28   302
               steak     6   275

POTENTIAL MEMORY LEAK: Not all dynamically-allocated memory explicitly freed/deleted.


<<EXPECTED>> (Different user prompt wording is okay.)

Enter a space-separated food, weight, and calories...
>>> blueberries 3 76
>>> sludge 1000 2000
>>> pho 28 302
>>> steak 6 275

         LUNCH MENU
ITEM          WEIGHT  CALORIES
apple           4       100
salad           2        80
blueberries     3        76
sludge       1000      2000
pho            28       302
steak           6       275


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

---------------------------- PURPOSE OF 2ND RUN ----------------------------
Verify a table of food properties.
------------------------- CODE CHANGES FOR 2ND RUN -------------------------
LUNCH_QTY = 2
----------------------------- START OF 2ND RUN -----------------------------

               Apple     4   100
               Salad     2    80
```

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

---------------------------- PURPOSE OF 3RD RUN ----------------------------
Verify a table of food properties.
-------------------------- CODE CHANGES FOR 3RD RUN --------------------------
LUNCH_QTY = 20
------------------------------ START OF 3RD RUN ------------------------------
THIS RUN FAILED BECAUSE:
   1. Items in the table are not properly aligned.
   2. A dynamic memory problem was encountered.


Input the space separated name, weight, and caloric content of a food item: blueberries
3 76
Input the space separated name, weight, and caloric content of a food item: pho 28 302
Input the space separated name, weight, and caloric content of a food item: steak 6 275
Input the space separated name, weight, and caloric content of a food item: tacos 10
249
Input the space separated name, weight, and caloric content of a food item: milk 7 215
Input the space separated name, weight, and caloric content of a food item: horseburger
12 934
Input the space separated name, weight, and caloric content of a food item: tequila 26
2418
Input the space separated name, weight, and caloric content of a food item: tripe 15
587
Input the space separated name, weight, and caloric content of a food item: salt 0 0
Input the space separated name, weight, and caloric content of a food item: cranberries
1 10
Input the space separated name, weight, and caloric content of a food item: ham 11 237
Input the space separated name, weight, and caloric content of a food item: gravy 2 446
Input the space separated name, weight, and caloric content of a food item: beans 11
198
Input the space separated name, weight, and caloric content of a food item: bread 4 98
Input the space separated name, weight, and caloric content of a food item: salmon 9
427
Input the space separated name, weight, and caloric content of a food item: avacado 3
187
Input the space separated name, weight, and caloric content of a food item: Gaejangguk
28 1449
Input the space separated name, weight, and caloric content of a food item: ants 10 233
               Apple     4   100
               Salad     2    80
         blueberries     3    76
                 pho    28   302
               steak     6   275
               tacos    10   249
                milk     7   215
         horseburger    12   934
             tequila    26  2418
               tripe    15   587
                salt     0     0
         cranberries     1    10
                 ham    11   237
               gravy     2   446
               beans    11   198
               bread     4    98
              salmon     9   427
             avacado     3   187
          Gaejangguk    28  1449
                ants    10   233
```

```
POTENTIAL MEMORY LEAK: Not all dynamically-allocated memory explicitly freed/deleted.


<<EXPECTED>> (Different user prompt wording is okay.)

Enter a space-separated food, weight, and calories...
>>> blueberries 3 76
>>> pho 28 302
>>> steak 6 275
>>> tacos 10 249
>>> milk 7 215
>>> horseburger 12 934
>>> tequila 26 2418
>>> tripe 15 587
>>> salt 0 0
>>> cranberries 1 10
>>> ham 11 237
>>> gravy 2 446
>>> beans 11 198
>>> bread 4 98
>>> salmon 9 427
>>> avacado 3 187
>>> Gaejangguk 28 1449
>>> ants 10 233

         LUNCH MENU
ITEM           WEIGHT  CALORIES
apple            4       100
salad            2        80
blueberries      3        76
pho             28       302
steak            6       275
tacos           10       249
milk             7       215
horseburger     12       934
tequila         26      2418
tripe           15       587
salt             0         0
cranberries      1        10
ham             11       237
gravy            2       446
beans           11       198
bread            4        98
salmon           9       427
avacado          3       187
Gaejangguk      28      1449
ants            10       233


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

----------------------------- PURPOSE OF 4TH RUN -----------------------------
Verify that program detects a memory allocation failure.
------------------------- CODE CHANGES FOR 4TH RUN -------------------------
Intentionally induced malloc failure.
------------------------------ START OF 4TH RUN ------------------------------
THIS RUN FAILED BECAUSE:
   1. Program did not exit when expected.  Does your program...
      ...prompt when it should not or not prompt when it should?
```

```
      ...wait for user input not mentioned in the requirements?
      ...re-run even if not stated in the requirements?
      ...hold the command window open before terminating?
      ...not terminate if a file open or memory allocation fails?
      ...get into an infinite loop?
   2. Program did not exit when expected.  Does your program...
      ...prompt when it should not or not prompt when it should?
      ...wait for user input not mentioned in the requirements?
      ...re-run even if not stated in the requirements?
      ...hold the command window open before terminating?
      ...not terminate if a file open or memory allocation fails?
      ...get into an infinite loop?


Input the space separated name, weight, and caloric content of a food item: blueberries
3 76
Memory allocation failed.
Input the space separated name, weight, and caloric content of a food item:
<<<<< PROGRAM SHOULD HAVE TERMINATED HERE BUT DID NOT >>>>>

                        <<<<< YOUR PROGRAM HUNG >>>>>

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