Consolidated Assignment 7 Report

This report contains the graded results for the newest of each exercise submitted to the assignment checker prior to 5/20/2020 2:42:16 PM PDT.

Student Name: Shaun Chemplavil
Student ID: U08713628
Contact e-mail: shaun.chemplavil@gmail.com
C/C++ Programming I (Section 146359)

Submitted:
   Exercise 0: 4/25/2020 11:44:35 AM PDT
   Exercise 1: 5/8/2020 8:37:02 AM PDT
   Exercise 2: 5/8/2020 8:37:23 AM PDT

Score (out of 20 possible): 16

```
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
"Announcements" page of the course website.  The assignment checker DOES NOT GRADE your
submissions but merely reports on issues so you can correct them and resubmit, thereby
avoiding unnecessary credit loss.  ALL GRADING IS DONE MANUALLY BY THE INSTRUCTOR after
the assignment deadline based solely upon the NEWEST submission of each exercise.  BE
WARY of correcting minor issues after the deadline because a late deduction will
usually be much greater than a minor issue deduction.


   From: Shaun Chemplavil <mailto:shaun.chemplavil@gmail.com>
   Subject: C1A7E0_U08713628
   Submitted: 4/25/2020 11:44:35 AM PDT
   Course: C/C++ Programming I (Section 146359)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 7, Exercise 0
   Exercise point value: 6
   File submitted:
      C1A7E0_Quiz.txt


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

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.  BE WARY of
correcting minor issues after the deadline because a late deduction will usually be
much greater than a minor issue deduction.
```

-3

```
Shaun Chemplavil U08713628
shaun.chemplavil@gmail.com
C/C++ Programming I : Fundamental Programming Concepts
146359 Raymond L. Mitchell, Jr., M.S.
04/25/2020
C1A7E0_Quiz.txt
Answers to Quiz

1. E
2. C <---E
3. E <---C
4. B
5. C <---E
6. B
```

```
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
"Announcements" page of the course website.  The assignment checker DOES NOT GRADE your
submissions but merely reports on issues so you can correct them and resubmit, thereby
avoiding unnecessary credit loss.  ALL GRADING IS DONE MANUALLY BY THE INSTRUCTOR after
the assignment deadline based solely upon the NEWEST submission of each exercise.  BE
WARY of correcting minor issues after the deadline because a late deduction will
usually be much greater than a minor issue deduction.


   From: Shaun Chemplavil <mailto:shaun.chemplavil@gmail.com>
   Subject: C1A7E1_U08713628
   Submitted: 5/8/2020 8:37:02 AM PDT
   Course: C/C++ Programming I (Section 146359)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 7, Exercise 1
   Exercise point value: 7
   Files submitted:
      C1A7E1_MyTime.h
      C1A7E1_main.cpp
      C1A7E1_DetermineElapsedTime.cpp

"Compile-time" results:
   No "compile-time" issues;
"Run-time" results:
   Program ran - No errors detected during preliminary testing (SEE ATTACHMENT);
```

```
1                                                                              80
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming I : Fundamental Programming Concepts
// 146359 Raymond L. Mitchell Jr.
// 05 / 08 / 2020
// C1A7E1_MyTime.h
// Win10
// Visual C++ 19.0
//
// This file contains the definition of the MyTime structure
//

#ifndef C1A7E1_MYTIME_H
#define C1A7E1_MYTIME_H
struct MyTime { int hours, minutes, seconds; };
#endif
```

```
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming I : Fundamental Programming Concepts
// 146359 Raymond L. Mitchell Jr.
// 05 / 08 / 2020
// C1A7E1_DetermineElapsedTime.cpp
// Win10
// Visual C++ 19.0
//
// This function calculates the elapsed time between two MyTime structure
//  variables
//

#include "C1A7E1_MyTime.h"

const int HOURS_TO_SECS = 3600;
const int MINUTES_TO_SECS = 60;
const long DAYS_TO_SECS = 86400;

MyTime *DetermineElapsedTime(const MyTime *startTime, const MyTime *stopTime)
{
   long startTimeSec, stopTimeSec, elaspedTimeSec;
   static MyTime elapsedTime;

   // Convert Start and Stop Time to seconds
   startTimeSec = (long)startTime->seconds;
   startTimeSec += (long)(startTime->minutes * MINUTES_TO_SECS);
   startTimeSec += (long)(startTime->hours * HOURS_TO_SECS);

   stopTimeSec = (long)stopTime->seconds;
   stopTimeSec += (long)(stopTime->minutes * MINUTES_TO_SECS);
   stopTimeSec += (long)(stopTime->hours * HOURS_TO_SECS);

   elaspedTimeSec = stopTimeSec - startTimeSec;

   // When elaspedTimeSec is negative, stopTime refers to next day
   // so we must add DAYS_TO_SECS
   if (elaspedTimeSec <= 0)
   {
      elaspedTimeSec += DAYS_TO_SECS;
   }

   // Format result for MyTime structure
   elapsedTime.hours = (int)(elaspedTimeSec / (long)HOURS_TO_SECS);

   // Remove seconds associated with the hours captured above
   elaspedTimeSec %= (long)HOURS_TO_SECS;
   elapsedTime.minutes = (int)(elaspedTimeSec / (long)MINUTES_TO_SECS);

   // Remove seconds associated with the minute captured above
   elaspedTimeSec %= (long)MINUTES_TO_SECS;
   elapsedTime.seconds = (int)elaspedTimeSec;

   return(&elapsedTime);
}
```

?

Possible overflow if int is 16 bits wide. Max 16-bit int value = 32767; Max possible seconds in this expression exceeds that value.

$\boxed{-1}$

None of these type long casts are necessary.

```
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming I : Fundamental Programming Concepts
// 146359 Raymond L. Mitchell Jr.
// 05 / 08 / 2020
// C1A7E1_main.cpp
// Win10
// Visual C++ 19.0
//
// This program will prompt the user to input MyTime structure variables
// and then output the elasped time between them
//

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

#include "C1A7E1_MyTime.h"

MyTime *DetermineElapsedTime(const MyTime *start, const MyTime *stop);

// Define the number of times we repeat the main body
const int REPEAT = 3;

int main()
{
   // Need to setfill, to properly display single digit hour/min/sec
   cout << setfill('0');

   // Repeat main Block REPEAT times
   for (int reps = 0; reps < REPEAT; reps++)
   {
      char delim;
      MyTime start, stop, *elapse;

      // Request and Store User Input
      cout <<
         "Enter the space separated stop and start time in HH:MM:SS format :\n";

      cin >> start.hours >> delim >> start.minutes >> delim >> start.seconds
         >> stop.hours >> delim >> stop.minutes >> delim >> stop.seconds;

      elapse = DetermineElapsedTime(&start, &stop);

      cout << "The time elapsed from "
         << setw(2) << start.hours << delim << setw(2) << start.minutes << delim
         << setw(2) << start.seconds
         << " to " << setw(2) << stop.hours << delim << setw(2) << stop.minutes
         << delim << setw(2) << stop.seconds
         << " is " << setw(2) << elapse->hours << delim << setw(2)
         << elapse->minutes << delim << setw(2) << elapse->seconds << "\n";
   }
   return 0;
}
```

```
********** 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 the space separated stop and start time in HH:MM:SS format :
00:00:00 00:00:00
The time elapsed from 00:00:00 to 00:00:00 is 24:00:00
Enter the space separated stop and start time in HH:MM:SS format :
12:12:12 13:12:11
The time elapsed from 12:12:12 to 13:12:11 is 00:59:59
Enter the space separated stop and start time in HH:MM:SS format :
13:12:11 12:12:12
The time elapsed from 13:12:11 to 12:12:12 is 23:00:01

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

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

Enter the space separated stop and start time in HH:MM:SS format :
00:00:00 23:59:58
The time elapsed from 00:00:00 to 23:59:58 is 23:59:58
Enter the space separated stop and start time in HH:MM:SS format :
23:59:58 00:00:00
The time elapsed from 23:59:58 to 00:00:00 is 00:00:02
Enter the space separated stop and start time in HH:MM:SS format :
22:03:27 18:18:49
The time elapsed from 22:03:27 to 18:18:49 is 20:15:22

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

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

Enter the space separated stop and start time in HH:MM:SS format :
12:23:34 12:23:35
The time elapsed from 12:23:34 to 12:23:35 is 00:00:01
Enter the space separated stop and start time in HH:MM:SS format :
10:00:59 00:18:22
The time elapsed from 10:00:59 to 00:18:22 is 14:17:23
Enter the space separated stop and start time in HH:MM:SS format :
09:09:09 02:23:35
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
"Announcements" page of the course website.  The assignment checker DOES NOT GRADE your
submissions but merely reports on issues so you can correct them and resubmit, thereby
avoiding unnecessary credit loss.  ALL GRADING IS DONE MANUALLY BY THE INSTRUCTOR after
the assignment deadline based solely upon the NEWEST submission of each exercise.  BE
WARY of correcting minor issues after the deadline because a late deduction will
usually be much greater than a minor issue deduction.


   From: Shaun Chemplavil <mailto:shaun.chemplavil@gmail.com>
   Subject: C1A7E2_U08713628
   Submitted: 5/8/2020 8:37:23 AM PDT
   Course: C/C++ Programming I (Section 146359)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 7, Exercise 2
   Exercise point value: 7
   File submitted:
      C1A7E2_main.c

"Compile-time" results:
   No "compile-time" issues;
"Run-time" results:
   Program ran - No errors detected during preliminary testing (SEE ATTACHMENT);
```

```c
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming I : Fundamental Programming Concepts
// 146359 Raymond L. Mitchell Jr.
// 05 / 08 / 2020
// C1A7E2_main.c
// Win10
// Visual C++ 19.0
//
// This program prompts the user to enter nutritional information for
// LUNCH_QTY number of food items contained in a 'lunch' Food structure array
// 2 items within this structure array have been initialized
//

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

#define LUNCH_QTY 5
#define STR_LENGTH 129
#define PRE_INIT_STRUCT 2

int main(void)
{
   struct Food
   {
      char *name;            /* "name" attribute of food */
      int weight, calories; /* "weight" and "calories" attributes of food */
   }lunches[LUNCH_QTY] = {{"apple", 4, 100}, {"salad", 2, 80}};

   // Populate uninitialized structure array elements
   for (int lunchCnt = PRE_INIT_STRUCT; lunchCnt < LUNCH_QTY; lunchCnt++)
   {
      char buffer[STR_LENGTH];

      // get the users strings
      printf("Enter the whitespace separated name, weight, and calories: ");
      scanf("%128s%d%d",
         buffer,
         &lunches[lunchCnt].weight,
         &lunches[lunchCnt].calories);

      //  find number of characters input by user
      size_t buffSize = strlen(buffer);

      // increment buffsize to account for null character
      buffSize++;

      // Allocate memory to place name within structure element
      if ((lunches[lunchCnt].name = (char *)malloc(buffSize)) == NULL)
      {
         fputs("Not enough memory for name\n", stderr);
         exit(EXIT_FAILURE);
      }
      memcpy(lunches[lunchCnt].name, buffer, buffSize);
   }

   // Display Results
   for (int lunchCnt = 0; lunchCnt < LUNCH_QTY; lunchCnt++)
   {
      printf("%-15s %5d %5d\n",
         lunches[lunchCnt].name,
         lunches[lunchCnt].weight,
         lunches[lunchCnt].calories);
   }

   // free dynamically allocated memory
   for (int lunchCnt = PRE_INIT_STRUCT; lunchCnt < LUNCH_QTY; lunchCnt++)
   {
      free(lunches[lunchCnt].name);
   }

   return 0;
}
```

```
********** C1 ASSIGNMENT 7 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.                 **************


----------------------------- PURPOSE OF 1ST RUN ----------------------------
Verify a table of food properties.
-------------------------- CODE CHANGES FOR 1ST RUN --------------------------
LUNCH_QTY = 6
------------------------------ START OF 1ST RUN ------------------------------

Enter the whitespace separated name, weight, and calories: blueberries 3 76
Enter the whitespace separated name, weight, and calories: sludge 1000 2000
Enter the whitespace separated name, weight, and calories: pho 28 302
Enter the whitespace separated name, weight, and calories: steak 6 275
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 ------------------------------

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

------------------------------- 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 ------------------------------

Enter the whitespace separated name, weight, and calories: blueberries 3 76
Not enough memory for name

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