Consolidated Assignment 2 Report

This report contains the graded results for the newest of each exercise submitted to the assignment checker prior to 7/22/2020 10:32:03 PM PDT.

Student Name: Shaun Chemplavil
Student ID: U08713628
Contact e-mail: shaun.chemplavil@gmail.com
C/C++ Programming II (Section 149123)

Submitted:
- Exercise 1: 7/18/2020 3:52:44 PM PDT
- Exercise 2: 7/18/2020 4:05:15 PM PDT
- Exercise 3: 7/20/2020 5:16:26 PM PDT
- Exercise 4: 7/21/2020 4:13:34 PM PDT

Score (out of 20 possible): 19.8

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 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: C2A2E1_U08713628
   Submitted: 7/18/2020 3:52:44 PM PDT
   Course: C/C++ Programming II (Section 149123)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 2, Exercise 1  (C2_001496735M02005X56496)
   Exercise point value: 3
   Files submitted:
      C2A2E1_CountIntBitsF.c
      C2A2E1_CountBitsM.h
      C2A2E1_main-Driver.c

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

```
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming II : Dynamic Memory and File I / O Concepts
// 149123 Raymond L.Mitchell, Jr., M.S.
// 07 / 18 / 2020
// C2A2E1_CountBitsM.h
// Win10
// Visual C++ 19.0
//
// Header File contains CountBitsM macro definition, which determines the
// number of bits the datatype passed to it takes
//

#ifndef C2A2E1_COUNTBITSM_H
#define C2A2E1_COUNTBITSM_H

#include <limits.h>

// Macro that determines the bits of storage for the input data type
#define CountBitsM(objectOrType) ((int)sizeof(objectOrType)*CHAR_BIT)

#endif
```

?

```
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming II : Dynamic Memory and File I / O Concepts
// 149123 Raymond L.Mitchell, Jr., M.S.
// 07 / 18 / 2020
// C2A2E1_CountIntBitsF.c
// Win10
// Visual C++ 19.0
//
// File contains CountIntBitsF function, which determines the number of bits
// an int datatype variable takes
//
// ANSWER: No, datatype sizes are constant during the same implementation
//

int CountIntBitsF(void)
{
   unsigned int sizeTemp = 1;
   int count;

   // Keep left shifting sizeTemp by 1 bit until sizeTemp = 0
   for (count = 0; sizeTemp != 0; count++)
   {
      sizeTemp <<= 1;
   }

   // When sizeTemp = 0 we have cycled through all bits of the variable
   // count contains the number of bits
   return(count);
}
```

Irrelevant

Your answer is not correct.

overscoped

```
********** C2 ASSIGNMENT 2 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 RUN --------------------------------

Implementation-dependent bit widths:

   From CountIntBitsF:
      Type 'int'         -> 32

   From CountBitsM:
      Type 'char'        -> 8
      Type 'short'       -> 16
      Type 'int'         -> 32
      Type 'long'        -> 32
      Type 'float'       -> 32
      Type 'double'      -> 64
      Type 'long double' -> 64

      printf return      -> 32
      'A'                -> 32
      2000000+8000000    -> 32
      xyz                -> 64
      cArray             -> 200
      dArray             -> 1600
      TEST               -> 192


--------------------------------- END OF 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 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: C2A2E2_U08713628
   Submitted: 7/18/2020 4:05:15 PM PDT
   Course: C/C++ Programming II (Section 149123)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 2, Exercise 2  (C2_001929781M02005X98929)
   Exercise point value: 5
   Files submitted:
      C2A2E2_main-Driver.cpp
      C2A2E2_CountIntBitsF.cpp
      C2A2E2_Rotate.cpp

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

```
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming II : Dynamic Memory and File I / O Concepts
// 149123 Raymond L.Mitchell, Jr., M.S.
// 07 / 18 / 2020
// C2A2E2_CountIntBitsF.cpp
// Win10
// Visual C++ 19.0
//
// File contains CountIntBitsF function, which determines the number of bits
// an int datatype variable takes
//

int CountIntBitsF()
{
   unsigned int sizeTemp = 1;
   int count;

   // Keep left shifting sizeTemp by 1 bit until sizeTemp = 0
   for (count = 0; sizeTemp != 0; count++)
   {
      sizeTemp <<= 1;
   }

   // When sizeTemp = 0 we have cycled through all bits of the variable
   // count contains the number of bits
   return(count);
}
```

```cpp
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming II : Dynamic Memory and File I / O Concepts
// 149123 Raymond L.Mitchell, Jr., M.S.
// 07 / 18 / 2020
// C2A2E2_Rotate.cpp
// Win10
// Visual C++ 19.0
//
// File containing definition of the Rotate function, which rotates the bits of
// the 'object' parameter, but the parameter define 'count' integer, rotation
// will preserve the original bits, but move their placement (so the least
// significant bit will be rotated to the most significant bit, and vice versa
// .... depending on the direction of the rotation)
//

int CountIntBitsF();

unsigned Rotate(unsigned object, int count)
{
   int leftShiftCnt, rightShiftCnt, numBits = CountIntBitsF();
   unsigned leftShiftObj, rightShiftObj;

   // avoid shifting by more bits contained in the object
   count %= numBits;

   // For most efficient Rotation, we need to check sign of count
   // this will determine how many counts to right/left shift
   if (count >= 0)
   {
      rightShiftCnt = count;
      leftShiftCnt = numBits - count;
   }
   else
   {
      leftShiftCnt = -count;
      rightShiftCnt = numBits + count;
   }

   rightShiftObj = object >> rightShiftCnt;
   leftShiftObj = object << leftShiftCnt;

   return(leftShiftObj | rightShiftObj);
}
```

```
********** C2 ASSIGNMENT 2 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 RUN --------------------------------

0x5 rotated right by 1 bit is 0x80000002

0x5 rotated left by 1 bit is 0xa

0x5 rotated right by 64 bits is 0x5

0x8765 rotated left by 64 bits is 0x8765

0x8765 rotated right by 3217 bits is 0x43b28000

0x8765 rotated left by 3217 bits is 0xeca0001


--------------------------------- END OF 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 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: C2A2E3_U08713628
Submitted: 7/20/2020 5:16:26 PM PDT
Course: C/C++ Programming II (Section 149123)
Student's name: Shaun Chemplavil
Contact email: shaun.chemplavil@gmail.com
Student ID: U08713628
Assignment 2, Exercise 3 (C2_005657653M02005X24657)
Exercise point value: 6
File submitted:
C2A2E3_StackFrames.pdf

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.

Shaun Chemplavil U08713628

shaun.chemplavil@gmail.com

C/C++ Programming II : Dynamic Memory and File I/O Concepts

149123 Raymond L. Mitchell, Jr., M.S.

07/20/2020

C2A2E3_StackFrames.pdf

This file contains the stack frame for the given "Greatest Common Divisor" recursive function shown in the prompt

-0.2

Relative Address	Absolute Address	Stack Value	Description
			startup
BP+7h	B8Fh	1005h	Function Return Address
BP	B88h	0h	Previous Frame Address
			Function main
~~**BP+Ch**~~	B83h	??	Return Object (int)
BP+7h	B7Ch	??	Function Return Address (pointer)
BP	B75h	B88h	Previous Frame Address (pointer)
BP-5h	B70h	??	result (int)
			Function ready
~~**BP+Fh**~~	B68h	??	Return Object (long – implied conversion from short)
BP+7h	B61h	62Dh	Function Return Address (pointer) (assignment to result)
BP	B5Ah	B75h	Previous Frame Address (pointer)
BP-4h	B56h	??	temp (short)
			Function gcd 1
BP+1Ah	B51h	??	Return Object (int – implied conversion from long)
BP+16h	B4Dh	360	y argument (short)
BP+Eh	B45h	480	x argument (long)
BP+7h	B3Eh	3F0h	Function Return Address (pointer) (assignment to **temp**)
BP	B37h	B5Ah	Previous Frame Address (pointer)
			Function gcd 2
BP+1Ah	B32h	??	Return Object (int – implied conversion from long)
BP+16h	B2Eh	360	y argument (short – implied conversion from long)
BP+Eh	B26h	120	x argument (long)
BP+7h	B1Fh	45Dh	Function Return Address (pointer) (return on line 42)
BP	B18h	B37h	Previous Frame Address (pointer)
			Function gcd 3
BP+1Ah	B13h	??	Return Object (int – implied conversion from long)
BP+16h	B0Fh	0	y argument (short - implied conversion from long)
BP+Eh	B07h	120	x argument (long)
BP+7h	B00h	45Dh	Function Return Address (pointer) (return on line 42)
BP	AF9h	B18h	Previous Frame Address (pointer)

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 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: C2A2E4_U08713628
   Submitted: 7/21/2020 4:13:34 PM PDT
   Course: C/C++ Programming II (Section 149123)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 2, Exercise 4  (C2_001593213M02005X45593)
   Exercise point value: 6
   Files submitted:
      C2A2E4_main-Driver.cpp
      C2A2E4_Reverse.cpp
      C2A2E4_OpenFile.cpp

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

```
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming II : Dynamic Memory and File I / O Concepts
// 149123 Raymond L.Mitchell, Jr., M.S.
// 07 / 21 / 2020
// C2A2E4_OpenFile.cpp
// Win10
// Visual C++ 19.0
//
// File contains OpenFile function
//
#include <iostream>
#include <fstream>

using namespace std;

static void ErrorAndQuit(const char *myString)
{
   cerr << "\"" << myString << "\" :File access error!\n";
   exit(-1);
}

void OpenFile(const char *fileName, ifstream &inFile)
{
   // open fileName in "read" mode using the ifstream object inFile
   inFile.open(fileName);
   if (!inFile.is_open())
      ErrorAndQuit(fileName);
}
```

Title block provides no meaningful information about the purpose/functionality of what is in this file.

Why clutter your code with a function that's only called once from one place?

```
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C / C++ Programming II : Dynamic Memory and File I / O Concepts
// 149123 Raymond L.Mitchell, Jr., M.S.
// 07 / 21 / 2020
// C2A2E4_Reverse.cpp
// Win10
// Visual C++ 19.0
//
// File contains Reverse function, it recursively reads in characters from a
// text file in inFile, until a separator (defined as whitespace .?!!,:; or EOF)
// is encountered, then the characters are displayed in reverse order with the
// last and next to next to last characters being displayed as capitalized
//

#include <iostream>
#include <fstream>
#include <cctype>

using namespace std;

// Define the "levels" that we want to force a capitalization
const int CAPLEVEL1 = 1;
const int CAPLEVEL2 = 3;

// Function to check if character is a 'separator'
inline bool isSeparator(int thisChar)
{
   return(isspace(thisChar) || thisChar == '.' || thisChar == '?' ||
      thisChar == '!' || thisChar == ',' || thisChar == ':' ||
      thisChar == ';' || thisChar == EOF);
}

int Reverse(ifstream &inFile, const int level)
{
   int thisChar;

   // Get Current Character
   thisChar = inFile.get();

   // Return character if it is a separator
   if (isSeparator(thisChar))
      return(thisChar);

   // if not a separator, continue reversing
   int thisSeparator;
   thisSeparator = Reverse(inFile, level + 1);

   //  Capitalize character if last or "next to next to last" character
   if (level == CAPLEVEL1 || level == CAPLEVEL2)
      cout << (char)toupper(thisChar);
   else
      cout << (char)thisChar;

   return thisSeparator;
}
```

Magic comment: Provides erroneous information if a different level is used.

Magic comment: Provides erroneous information if a different level is used.

```
********** C2 ASSIGNMENT 2 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.                  **************


----------------------------- PURPOSE OF 1ST RUN -----------------------------
Verify string reversal.
-------------------------- CODE CHANGES FOR 1ST RUN --------------------------
Using input file "TestFile2.txt"
------------------------------ START OF 1ST RUN ------------------------------

tXeT eLiF elpmAxE:  hCaE droCeR sI EnO eNiL fO iBrA,T,yRaR htgNeL sdNe( hTiW )'N\':
tAhW!  rehtOnA sselEsU, dipUtS, DnA yrasseceNnU margOrP?
SeY;  tAhW eSlE?:  YrT tuPnI R.E,D:I;R?E!n\o/i*T=C.  */./*  */.!?,;:/*+=
*/ gnIsU C /*
enifEd# ON_CER 9*/ rebMuN fO droCeR oT dAeR /*

rAhC HGUONE_GIB[pMeT?]; */ yaRrA RoF LoH;gNiD YnA droCeR /*
TnI I;

RoF i( = 0; I < ON_CER; )I++*/ RoF tsRiF dedeeNnU sdroCeR /*
fI cSf(?pf(FnA, )"c*%]n\^[*%" == )FOE {   */ dAeR DnA woRhT yAwA /*
detcepxenU"(stUpF "n\FOE, )rreDtS;   */ erEhT sI oN CeR.DrO ON_CER /*
)ERULIAF_TIXE(tIxE;*/ roRrE tIxE /*
}
fI Gf(:pmet(StE, )pmet(foeZiS, )pF == )LLUN {*/ dAeR droCeR ON_CER /*
pxenU"(stUpF!deTcE "n\FOE, )rreDtS;   */ erEhT sI oN CeR.DrO ON_CER /*
)ERULIAF_TIXE(tIxE;   */ roRrE tIxE /*
}
*/ tA sIhT tnIoP (?) droCeR # ON_CER sI nI pMeT. /*




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

----------------------------- PURPOSE OF 2ND RUN -----------------------------
Verify string reversal.
-------------------------- CODE CHANGES FOR 2ND RUN --------------------------
Using input file "TestFile11.txt"
------------------------------ START OF 2ND RUN ------------------------------

ON_CER; )I++ */ oL O P hguoRhT eNnU E DeD sdroCeR POOL /*
 fI cSf(?pf(FnA, )"c*%]n\^[*%" == )FOE  { */ eS E D & oRhT W yAwA /*




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

----------------------------- PURPOSE OF 3RD RUN -----------------------------
Verify that program detects an input file open failure.
-------------------------- CODE CHANGES FOR 3RD RUN --------------------------
Using input file "bad//file//a"
------------------------------ START OF 3RD RUN ------------------------------
```

```
"bad//file//a" :File access error!



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

----------------------------- PURPOSE OF 4TH RUN -----------------------------
Verify that program detects an input file open failure.
-------------------------- CODE CHANGES FOR 4TH RUN --------------------------
Using input file "bad//file//b"
------------------------------- START OF 4TH RUN -------------------------------

"bad//file//b" :File access error!



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