Consolidated Assignment 5 Report

This report contains the graded results for the newest of each exercise submitted to the assignment checker prior to 8/12/2020 10:03:58 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: 8/2/2020 1:58:58 PM PDT
   Exercise 2: 8/5/2020 4:01:38 AM PDT
   Exercise 3: 8/9/2020 12:37:36 PM PDT
   Exercise 4: 8/10/2020 3:39:03 AM PDT
```

Score (out of 20 possible): 20

```
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: C2A5E1_U08713628
   Submitted: 8/2/2020 1:58:58 PM PDT
   Course: C/C++ Programming II (Section 149123)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 5, Exercise 1  (C2_00158635M02005X37058)
   Exercise point value: 4
   Files submitted:
      C2A5E1_main-Driver.c
      C2A5E1_SwapObjects.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.
// 08 / 02 / 2020
// C2A5E1_SwapObjects.c
// Win10
// Visual C++ 19.0
//
// File contains SwabObjects function, which swaps the objects from one pointer
// address to the other
//

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

void SwapObjects(void *pa, void *pb, size_t size)
{
   void *temp;

   // Allocate appropriate memory
   if ((temp = malloc(size)) == NULL)
   {
      fputs("Out of memory\n", stderr);
      exit(EXIT_FAILURE);
   }

   // Copy object at pa to temp
   memcpy(temp, pa, size);
   // Copy object at pb to pa
   memcpy(pa, pb, size);
   // Copy object at temp (original object at pa) to pb
   memcpy(pb, temp, size);
   // free allocated memory
   free(temp);
}
```

```
********** C2 ASSIGNMENT 5 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.                 *************


------------------------------ PURPOSE OF 1ST RUN ------------------------------
Verify that objects were swapped.
------------------------------- START OF 1ST RUN -------------------------------

SwapObjects succeeded



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

------------------------------ PURPOSE OF 2ND RUN ------------------------------
Verify that program detects a memory allocation failure.
-------------------------- CODE CHANGES FOR 2ND RUN ---------------------------
Intentionally induced malloc failure.
------------------------------- START OF 2ND RUN -------------------------------

Out of memory



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

```
   From: Shaun Chemplavil <mailto:shaun.chemplavil@gmail.com>
   Subject: C2A5E2_U08713628
   Submitted: 8/5/2020 4:01:38 AM PDT
   Course: C/C++ Programming II (Section 149123)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 5, Exercise 2  (C2_001485264M02005X60485)
   Exercise point value: 6
   Files submitted:
      C2A5E2_Create2D.c
      C2A5E2_Type-Driver.h
      C2A5E2_main-Driver.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 II : Dynamic Memory and File I / O Concepts
// 149123 Raymond L.Mitchell, Jr., M.S.
// 08 / 05 / 2020
// C2A5E2_Create2D.c
// Win10
// Visual C++ 19.0
//
// File contains Create2D  and Free2D functions,
//  Create2D creates a 2-d pointer array of data type Type, the dimensions are
//  specified via input arguments.  Free2D frees the data at the pointer passed
// to it
//

#include <stdio.h>
#include <stdlib.h>
#include "C2A5E2_Type-Driver.h"

Type **Create2D(size_t rows, size_t cols)
{
   Type **p, **p1, **end;

   size_t typeSize = sizeof(Type);

   // Allocate appropriate memory
   if ((p = malloc(sizeof(Type *) * rows + typeSize * rows * cols)) == NULL)
   {
      fputs("Out of memory\n", stderr);
      exit(EXIT_FAILURE);
   }
   for (end = p + rows, p1 = p; p1 < end; ++p1)
   {
      // find distance from original pointer to next element of pointer array
      size_t rowNum = (size_t)(p1 - p);

      //Calculate address of the start of the next row and place within pointer
      // array, we must typecast to size_t to avoid pointer math which ensures
      // the size of Type is taken into account correctly
      *p1 = (Type *)((size_t)end + typeSize * (cols * rowNum));
   }

   return(p);
}

void Free2D(void *p)
{
   // because all memory was allocated at one tine, we can free everything
   free(p);
}
```

```
********** C2 ASSIGNMENT 5 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 -----------------------------
Test pointer array creation.
------------------------------ START OF 1ST RUN ------------------------------

Create2D(1, 27) succeeded
Create2D(2, 26) succeeded
Create2D(3, 25) succeeded
Create2D(4, 24) succeeded
Create2D(5, 23) succeeded
Create2D(6, 22) succeeded
Create2D(7, 21) succeeded
Create2D(8, 20) succeeded
Create2D(9, 19) succeeded
Create2D(10, 18) succeeded
Create2D(11, 17) succeeded
Create2D(12, 16) succeeded
Create2D(13, 15) succeeded
Create2D(14, 14) succeeded
Create2D(15, 13) succeeded
Create2D(16, 12) succeeded
Create2D(17, 11) succeeded
Create2D(18, 10) succeeded
Create2D(19, 9) succeeded
Create2D(20, 8) succeeded
Create2D(21, 7) succeeded
Create2D(22, 6) succeeded
Create2D(23, 5) succeeded
Create2D(24, 4) succeeded
Create2D(25, 3) succeeded
Create2D(26, 2) succeeded



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

----------------------------- PURPOSE OF 2ND RUN -----------------------------
Verify that program detects a memory allocation failure.
--------------------------- CODE CHANGES FOR 2ND RUN --------------------------
Intentionally induced malloc failure.
------------------------------ START OF 2ND RUN ------------------------------

Out of memory



------------------------------- END OF 2ND 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: C2A5E3_U08713628
   Submitted: 8/9/2020 12:37:36 PM PDT
   Course: C/C++ Programming II (Section 149123)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 5, Exercise 3  (C2_002745341M02005X98745)
   Exercise point value: 5
   File submitted:
      C2A5E3_StateDiagram.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

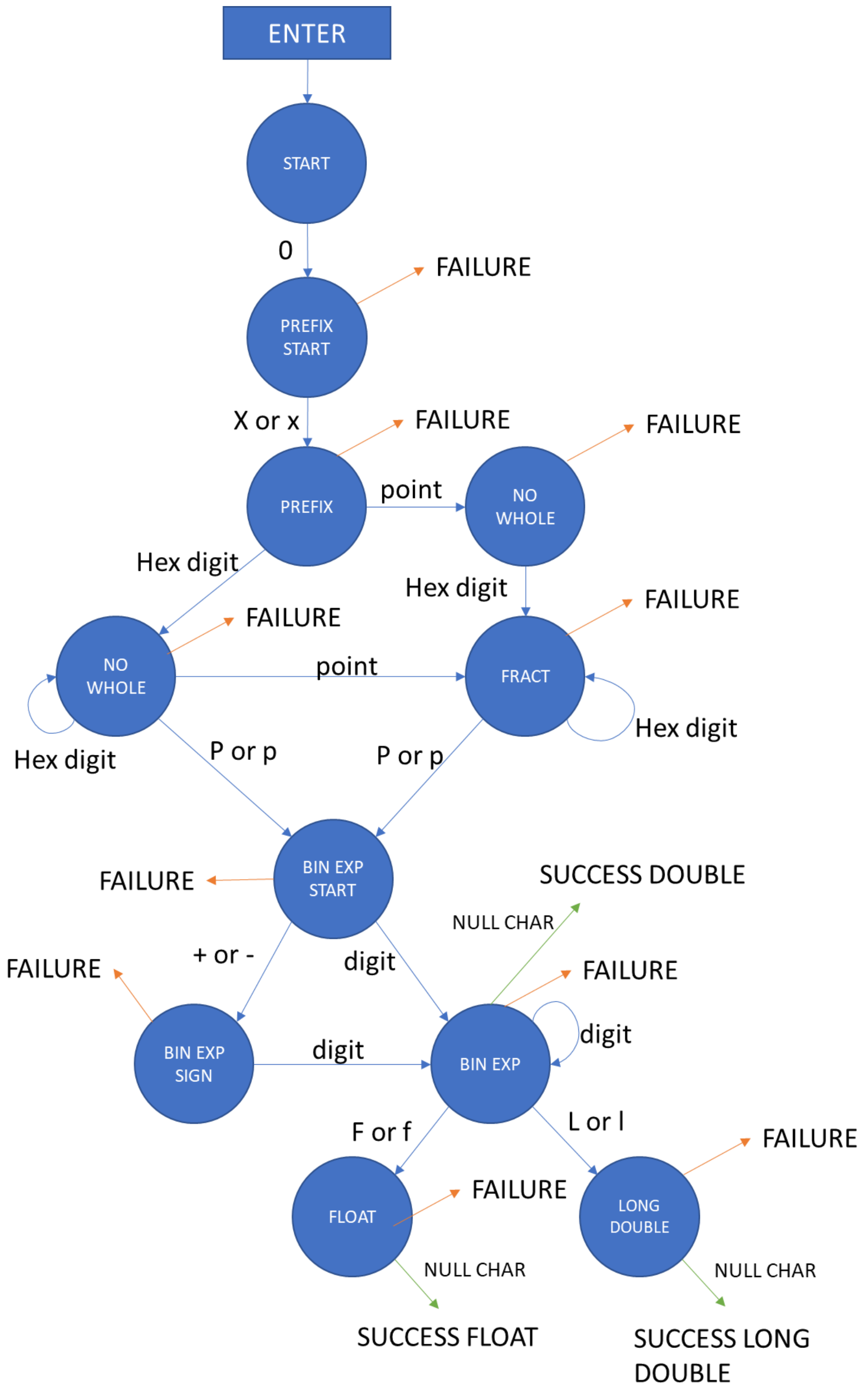
C/C++ Programming II : Dynamic Memory and File I/O Concepts

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

08/09/2020

C2A5E3_StateDiagram.pdf

This file contains a diagram for the State Machine representing the “DetectFloats’ function



```
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: C2A5E4_U08713628
   Submitted: 8/10/2020 3:39:03 AM PDT
   Course: C/C++ Programming II (Section 149123)
   Student's name: Shaun Chemplavil
   Contact email: shaun.chemplavil@gmail.com
   Student ID: U08713628
   Assignment 5, Exercise 4  (C2_002265407M02005X54265)
   Exercise point value: 5
   Files submitted:
      C2A5E4_StatusCode-Driver.h
      C2A5E4_OpenFile.cpp
      C2A5E4_main-Driver.cpp
      C2A5E4_DetectFloats.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.
// 08 / 10 / 2020
// C2A5E4_DetectFloats.cpp
// Win10
// Visual C++ 19.0
//
// File contains DetectFloats function, which determines if the string at chPtr
// represents a syntactically legal hexadecimal floating literal, and returns
// its data type,  This is architected via a state machine
//

#include <fstream>
#include "C2A5E4_StatusCode-Driver.h"

using namespace std;

StatusCode DetectFloats(const char *chPtr)
{
   // Define State Variable
   enum States
   {
      START, PREFIX_START, PREFIX, NO_WHOLE, WHOLE, FRACT, BIN_EXP_START,
      BIN_EXP_SIGN, BIN_EXP, FLOAT, LONG
   } state = START;

   do
   {
      // Enter Current State
      switch (state)
      {
         case START:
            switch (*chPtr++)
            {
               case '0':
                  state = PREFIX_START;
                  break;

               default:
                  return(NO_MATCH);
            }
            break;

         case PREFIX_START:
            switch (*chPtr++)
            {
               case 'X': case 'x':
                  state = PREFIX;
                  break;

               default:
                  return(NO_MATCH);
            }
            break;

         case PREFIX:
            switch (*chPtr++)
            {
               // HEX DIGIT
               case '0': case '1': case '2':case '3': case '4': case '5':
               case '6': case '7': case '8':case '9': case 'A': case 'B':
               case 'C': case 'D': case 'E':case 'F':case 'a': case 'b':
               case 'c': case 'd': case 'e':case 'f':
                  state = WHOLE;
                  break;

               case '.':
                  state = NO_WHOLE;
                  break;

               default:
                  return(NO_MATCH);
            }
            break;

         case  NO_WHOLE:
            switch (*chPtr++)
            {
               // HEX DIGIT
               case '0': case '1': case '2':case '3': case '4': case '5':
               case '6': case '7': case '8':case '9': case 'A': case 'B':
               case 'C': case 'D': case 'E':case 'F':case 'a': case 'b':
               case 'c': case 'd': case 'e':case 'f':
                  state = FRACT;
                  break;

               case 'P':case 'p':
                  state = BIN_EXP_START;
                  break;

               default:
                  return(NO_MATCH);
            }
            break;

         case  WHOLE:
            switch (*chPtr++)
            {
               // HEX DIGIT
               case '0': case '1': case '2':case '3': case '4': case '5':
               case '6': case '7': case '8':case '9': case 'A': case 'B':
               case 'C': case 'D': case 'E':case 'F':case 'a': case 'b':
               case 'c': case 'd': case 'e':case 'f':
                  state = WHOLE;
                  break;

               case '.':
                  state = FRACT;
                  break;

               case 'P':case 'p':
                  state = BIN_EXP_START;
                  break;

               default:
                  return(NO_MATCH);
            }
            break;

         case  FRACT:
            switch (*chPtr++)
            {
               // HEX DIGIT
               case '0': case '1': case '2':case '3': case '4': case '5':
               case '6': case '7': case '8':case '9': case 'A': case 'B':
               case 'C': case 'D': case 'E':case 'F':case 'a': case 'b':
               case 'c': case 'd': case 'e':case 'f':
                  state = FRACT;
                  break;

               case 'P':case 'p':
                  state = BIN_EXP_START;
                  break;

               default:
                  return(NO_MATCH);
            }
            break;

         case  BIN_EXP_START:
            switch (*chPtr++)
            {
               // DIGIT
               case '0': case '1': case '2':case '3': case '4': case '5':
               case '6': case '7': case '8':case '9':
                  state = BIN_EXP;
                  break;

               case '+': case '-':
                  state = BIN_EXP_SIGN;
                  break;

               default:
                  return(NO_MATCH);
            }
            break;

         case  BIN_EXP_SIGN:
            switch (*chPtr++)
            {
               // DIGIT
               case '0': case '1': case '2':case '3': case '4': case '5':
               case '6': case '7': case '8':case '9':
                  state = BIN_EXP;
                  break;

               default:
                  return(NO_MATCH);
            }
            break;

         case  BIN_EXP:
            switch (*chPtr++)
            {
               // DIGIT
               case '0': case '1': case '2':case '3': case '4': case '5':
               case '6': case '7': case '8':case '9':
                  state = BIN_EXP;
```

```
                     break;

                  case '\0':
                     return(TYPE_DOUBLE);

                  case 'L':case 'l':
                     state = LONG;
                     break;

                  case 'F':case 'f':
                     state = FLOAT;
                     break;

                  default:
                     return(NO_MATCH);
               }
               break;

            case FLOAT:
               switch (*chPtr++)
               {
                  case '\0':
                     return(TYPE_FLOAT);

                  default:
                     return(NO_MATCH);
               }

            case LONG:
               switch (*chPtr++)
               {
                  case '\0':
                     return(TYPE_LDOUBLE);

                  default:
                     return(NO_MATCH);
               }
      }
   } while (chPtr);

   // Unexpected condition
   return(NO_MATCH);
}
```

```
//
// 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.
// 08 / 10 / 2020
// C2A5E4_OpenFile.cpp
// Win10
// Visual C++ 19.0
//
// File contains OpenFile function, which opens the file (specified via a
// pointer to the name of filename) in read mode , use the ifstream reference
//  which is also passed to it
//

#include <iostream>
#include <fstream>

using namespace std;

void OpenFile(const char *fileName, ifstream &inFile)
{
   // open fileName in "read" mode using the ifstream object inFile
   inFile.open(fileName);
   if (!inFile.is_open())
   {
      {
         cerr << "\"" << fileName << "\" :File access error!\n";
         exit(-1);
      }
   }
}
```

```
********** C2 ASSIGNMENT 5 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 hexadecimal floating point format recognition.
-------------------------- CODE CHANGES FOR 1ST RUN --------------------------
Using input file "TestFile5.txt"
------------------------------ START OF 1ST RUN ------------------------------

                  ZZZZZZZ
                   12.35F
                     6E6f
                    6e-6f
                     .29F
                  1.e-35f
                456.2E-0F
                       0.
                       .0
                      0.0
                    3.0e0
                 0.0E-300
                   0.0e+0
                     0e-0
                   29.28L
                   554e0l
                     0E0L
                    2e+2L
             001.100E-001L
              0x15.920p-0f <--- hexadecimal float
                   0X.3P6F <--- hexadecimal float
               0xa.bCDP20f <--- hexadecimal float
           0X6Ab.CDep+591f <--- hexadecimal float
               0xEF.p-98F <--- hexadecimal float
                   0X3P90F <--- hexadecimal float
               0x015.020p0 <--- hexadecimal double
                   0X.00P6 <--- hexadecimal double
               0xab.CDp20 <--- hexadecimal double
             0x6Ab.CDep+59 <--- hexadecimal double
                 0xF.p-98 <--- hexadecimal double
                   0XAP23 <--- hexadecimal double
                 0X396P-78 <--- hexadecimal double
                 0XFFFFp0L <--- hexadecimal long double
               0X.00P-426l <--- hexadecimal long double
          0xbb.122bb3p-20l <--- hexadecimal long double
                   0x2p2L <--- hexadecimal long double
                 0x0.p-98L <--- hexadecimal long double
              0o17.26e-89f
                   0O.3E6F
                  0o7.120f
               0O.5e+591f
                 0o0.e-98F
                   0O3E90F
                    0o15.
                     0O0.
```

```
     0o.47e20
     0o6.e+59
     0o3.e-98
        0O2E3
    0O316E-78
        0O.0L
       0O0E0L
0o26.75e-99l
       0o2e2l
    0O0.e-98L
         -2.3
        +6.90
        +4.5L
    -0x5.4p2F
    +0xF.4p-2
     0xL.4p-2
    -0o5.4e2L
    +0o7.4e-2
     0o7.8e2F
        0o1.9
        3.3FF
      0o1.1LL
    0x1.1p0LL
   3.3e+44+3F
  0o3.3e+44F+
  0x3.3p+44F+
      0o1.1LL
    0x1.1p0LL
    3.3e+44F+
  0x3.3p+44F+
            7
            0
           0L
           0F
            0
         0x0L
         0o0F
           0u
       6.5L5L
     0x6.5L5L
     0o6.5L5L
      6.5e3e3
    0o6.5e3e3
    0x6.5p3p3
      0x8Pf-0
       0x8e-0
       0o8e-0
       24.6FL
         x9.6
         o2.3
           6e
         985L
     68.88Le6
        22.eL
         .e+8
       5.77L6
         2f2e
        38.9E
        6.2F6
          6e+
                    5.7E-
                    2.3-5
                 18.4E+0LL
                   4E+0F-
                  69.5E+-8

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

----------------------------- PURPOSE OF 2ND RUN -----------------------------
Verify that program detects an input file open failure.
--------------------------- CODE CHANGES FOR 2ND RUN ---------------------------
Using input file "bad//file//a"
------------------------------ START OF 2ND RUN ------------------------------

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



------------------------------- 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//b"
------------------------------ START OF 3RD RUN ------------------------------

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



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