

**Exercise 1 (10 points – C++ Program)**

\*\*\*\*\* FILE C2A8E1\_OpenFiles.cpp \*\*\*\*\*

```
//  
// ...the usual title block Student/Course/Assignment/Compiler information goes here...  
//  
// This file contains function OpenFiles, which opens one or more specified  
// files.  
//
```

```
#include <cstdlib>  
#include <fstream>  
#include <iostream>  
using std::cerr;  
using std::exit;  
using std::ifstream;  
using std::nothrow;
```

```
//  
// Use separate ifstream objects to open the number of files specified by  
// <count> having the names specified in <fileNames>. Each ifstream object  
// is kept in a dynamically-allocated array and a pointer to the first element  
// of that array is returned. If a file fails to open close all previously  
// opened files, output an error message, and terminate the program with  
// an error code.  
//
```

```
ifstream *OpenFiles(char * const fileNames[], size_t count)  
{  
    if (count == 0)  
    {  
        cerr << "No files specified!\n";           // syntax error message  
        exit(EXIT_FAILURE);                       // error termination  
    }  
  
    ifstream *files;  
    if ((files = new (nothrow) ifstream[count]) == NULL) // array of ifstream  
    {  
        cerr << "new out of memory\n";  
        exit(EXIT_FAILURE);                       // error termination  
    }  
  
    for (size_t which = 0; which < count; ++which)    // for each file  
    {  
        files[which].open(fileNames[which]);          // open the file  
        if (!files[which].is_open())                 // test the open  
        {  
            cerr << "File \"" << fileNames[which] << "\" didn't open.\n";  
            for (size_t ix = 0; ix < which; ++ix)    // for each open file  
                files[ix].close();                  // close the file  
            delete[] files;                          // delete dynamic alloc  
            exit(EXIT_FAILURE);                      // error termination  
        }  
    }  
    return files;  
}
```

----- EXERCISE CONTINUES ON NEXT PAGE -----

```
1
2 ***** FILE C2A8E1_MergeAndDisplay.cpp *****
3 //
4 // ...the usual title block Student/Course/Assignment/Compiler information goes here...
5 //
6 // This file contains function MergeAndDisplay, which displays the lines from
7 // one or more text files interleaved with each other.
8 //
9
10 #include <fstream>
11 #include <iostream>
12 using std::cout;
13 using std::ifstream;
14
15 //
16 // For each of the <count> text files in <files> read and display the first
17 // line from the first file, the first line from the second file, the first
18 // line from the third file, etc. Then read and display the second line from
19 // the first file, the second line from the second file, the second line from
20 // the third file, etc. Continue this process until all lines from all files
21 // have been displayed. If a file runs out of lines simply close that file
22 // and continue on with the remaining files.
23 //
24 void MergeAndDisplay(ifstream files[], size_t count)
25 {
26     const int BUFSIZE = 512;                // size of input buffer
27
28     for (size_t openFiles = count; openFiles;) // while any file is open
29     {
30         for (size_t ix = 0; ix < count; ++ix) // for each file
31         {
32             if (files[ix].is_open())          // is unread data in file
33             {
34                 char buf[BUFSIZE];            // input buffer
35                 if (!files[ix].getline(buf, BUFSIZE)) // read file line; chk EOF
36                 {
37                     files[ix].close();        // close the file
38                     --openFiles;              // reduce open file count
39                 }
40                 else
41                     cout << buf << '\n';      // read a line from file
42                                     // write line to output
43             }
44         }
45     }
```

C2A8E1 Screen Shots are on the next page...

## C2A8E1 Screen Shots

Test 1:

```
1 / 1
2 / 1
4 / 1
5 / 1
1 / 2
2 / 2
4 / 2
5 / 2
1 / 3
2 / 3
4 / 3
5 / 3
1 / 4
2 / 4
4 / 4

1 / 5
2 / 5
4 / 5
1 / 6
2 / 6
4 / 6
1 / 7
2 / 7
4 / 7
1 / 8
2 / 8
4 / 8
1 / 9
2 / 9
4 / 9
1 / 10

4 / 10
1 / 11
4 / 11
1 / 12
4 / 12
1 / 13
4 / 13
1 / 14

1 / 15
1 / 16
1 / 17
1 / 18
```

Test 2:

```
2 / 1
1 / 1
2 / 2
1 / 2
2 / 3
1 / 3
2 / 4
1 / 4
2 / 5
1 / 5
2 / 6
1 / 6
2 / 7
1 / 7
2 / 8
1 / 8
2 / 9
1 / 9

1 / 10
1 / 11
1 / 12
1 / 13
1 / 14
1 / 15
1 / 16
1 / 17
1 / 18
```

**Exercise 2 (10 points – C Program)**

```
1  ***** FILE C2A8E2_OpenFileBinary.c *****
2
3  /*
4   * ...the usual title block Student/Course/Assignment/Compiler information goes here...
5   *
6   * This file contains function OpenFileBinary, which opens the specified file
7   * in the binary read-only mode.
8   */
9
10
11 #include <stdio.h>
12 #include <stdlib.h>
13
14 /*
15  * Open the file named in <fileName> in the "read only" binary mode and return
16  * its FILE pointer if the open succeeds. If it fails display an error message
17  * and terminate the program with an error code.
18  */
19 FILE *OpenFileBinary(const char *fileName)
20 {
21     FILE *inFile;
22
23     /* Open the file named in <fileName> in the read-only binary mode. */
24     if ((inFile = fopen(fileName, "rb")) == NULL)
25     {
26         /* Print an error message and terminate with an error code. */
27         fprintf(stderr, "File \"%s\" didn't open.\n", fileName);
28         exit(EXIT_FAILURE);
29     }
30     return inFile;
31 }
32
33 ***** FILE C2A8E2_DisplayModifiedSingleReals.c *****
34
35 /*
36  * ...the usual title block Student/Course/Assignment/Compiler information goes here...
37  *
38  * This file contains functions:
39  *   DisplayModifiedSingleReal: Reads groups of 4 bytes from an already open
40  *   binary file and stores each group in an integer that's at least 32
41  *   bits wide. The file bytes are assumed to be in big endian order;
42  *   DisplayModifiedSingleReals: Interprets the least significant 32-bits of
43  *   its integer parameter as floating point value in "Modified Single
44  *   Real" format; displays the bit pattern and interpreted value.
45  */
46
47 #include <limits.h>
48 #include <math.h>
49 #include <stdio.h>
50
51 #define MASK_USED_BITS 0xFFFFFFFFuL /* mask over all used bits */
52 #define SIGN_MASK 0x80000000uL /* sign field bit mask */
53 #define EXP_MASK 0x7FC00000uL /* exponent field bit mask */
54 #define FRAC_MASK 0x003FFFFFuL /* fraction field bit mask */
55 #define FRAC_BITS 22 /* bits in fraction field */
56 #define EXP_NBIAS 255 /* normalized number exponent bias */
57 #define EXP_DBIAS 254 /* denormalized number exponent bias */
58 #define EXP_MAX 511 /* exponent maximum value */
```

```
1
2 #define PATTERN_BYTES 4          /* # of bytes to analyze per pattern */
3 #define BASE 2.0                 /* radix of representation */
4
5 /*
6  * Analyze the bit pattern in <pattern>, treating it as the representation
7  * of a "Modified Single Real" as described in the requirements for this
8  * exercise. If the pattern represents either a normalized number, a
9  * denormalized number, or a zero, display which one of those it is
10  * along with the numeric value it represents. If the pattern represents
11  * an infinity or a NAN simply display which one of those it is.
12  */
13 static void DisplayModifiedSingleReal(unsigned long pattern)
14 {
15     int isNegative = !(SIGN_MASK & pattern);      /* neg/pos */
16     int exponent = (EXP_MASK & pattern) >> FRAC_BITS; /* exponent field val */
17     long fraction = FRAC_MASK & pattern;          /* fraction field val */
18
19     printf("%s", isNegative ? "-" : "+");          /* display the sign */
20
21     if (exponent == 0 && fraction == 0)            /* test for zero */
22         printf("%e Zero\n", 0.0);
23     else if (exponent == EXP_MAX && fraction == 0) /* test for infinity */
24         printf("INF\n");
25     else if (exponent == EXP_MAX && fraction != 0) /* test for NAN */
26         printf("NAN\n");
27     else                                           /* is norm/denorm */
28     {
29         double result = fraction * pow(BASE, -FRAC_BITS); /* fract. part val */
30         if (exponent != 0)                             /* if is norm */
31         {
32             ++result;                                   /* add implicit 1 */
33             result *= pow(BASE, exponent - EXP_NBIAS); /* calc mantissa */
34             printf("%e Normal\n", result);
35         }
36         else                                           /* else is denorm */
37         {
38             result *= pow(BASE, exponent - EXP_DBIAS); /* calc mantissa */
39             printf("%e Denormal\n", result);
40         }
41     }
42 }
43
44 /*
45  * Read successive 32-bit patterns from the file in <inFile>. Each
46  * of these patterns represents a "Modified Single Real" as described
47  * in the requirements for this exercise and is stored in the file in
48  * big endian order. Call function DisplayModifiedSingleReal for each
49  * pattern to display its format (normalized, denormalized, etc.), and
50  * also its value if it is not an infinity or a NAN.
51  */
52 void DisplayModifiedSingleReals(FILE *inFile)
53 {
54     for (;;)
55     {
56         /* Array to hold pattern bytes. */
57         unsigned char patternArray[PATTERN_BYTES];
```

```
1      /*
2      * Read the next pattern. This cannot be placed directly into a scalar
3      * type due to the implementation-dependent widths of such types and
4      * because this algorithm must handle the big endian data from the
5      * file independent of the endianness of the machine running this code.
6      */
7      size_t bytesRead = fread(patternArray, 1, PATTERN_BYTES, inFile);
8      /* If complete pattern was read. */
9      if (bytesRead == PATTERN_BYTES)
10     {
11         unsigned long result = 0;
12         int byteIx, shiftIx = PATTERN_BYTES - 1;
13         /* Shift each byte into the correct position in result */
14         for (byteIx = 0; byteIx < PATTERN_BYTES; ++byteIx, --shiftIx)
15             result |= patternArray[byteIx] << (CHAR_BIT * shiftIx);
16         printf("0x%08lx ", result & MASK_USED_BITS);
17         DisplayModifiedSingleReal(result);
18     }
19     /* Else, partial or no pattern was read. */
20     else
21     {
22         if (bytesRead > 0)
23             printf("Unexpected EOF\n");
24         break;
25     }
26 }
27 }
```

C2A8E2 Screen Shots are on the next page...

## C2A8E2 Screen Shots

TestFile7.bin

```

C:\UCSD Courses\Common\...
0xffc00001 -NAN
0x7fffffff +NAN
0xffc00000 -INF
0x7fc00000 +INF
0xffbffffff -1.157921e+77 Normal
0x7fbffffff +1.157921e+77 Normal
0x80400000 -3.454467e-77 Normal
0x00400000 +3.454467e-77 Normal
0x80000001 -8.236092e-84 Denormal
0x003ffffff +3.454467e-77 Denormal
0x000d0a00 +7.037971e-78 Denormal
0x80000000 -0.000000e+00 Zero
0x00400000 +3.454467e-77 Normal
0x00000000 +0.000000e+00 Zero
0x00000000 +0.000000e+00 Zero
0x00000000 +0.000000e+00 Zero
0xd0000000 -3.689349e+19 Normal
0x0e000000 +1.244603e-60 Normal
0x000eba1f +7.949073e-78 Denormal
0x21cd09b4 +9.055762e-37 Normal
0x44206e69 +1.974913e+05 Normal
0x6d20534f +4.613071e+54 Normal
0x2e65646f +1.341916e-21 Normal
0xb40a0d0d -8.221310e-15 Normal
0x00000d0a +2.749208e-80 Denormal
0xba037d00 -1.257067e-07 Normal
0x54623958 +7.421596e+24 Normal
0x5462390b +7.421507e+24 Normal
0xffdabcd8 -NAN
0x7fe02ac6 +NAN
0xffc00000 -INF
0x7fc00000 +INF
0x5462380b +7.421212e+24 Normal
0x6369520b +9.174708e+42 Normal
0x54623968 +7.421614e+24 Normal
0x0000000b +9.059701e-83 Denormal
0x50000000 +3.689349e+19 Normal
0x014c0045 +6.563579e-76 Normal
0x0e7f0005 +4.939521e-60 Normal
0x00003a54 +1.229813e-79 Denormal
0x80000000 -0.000000e+00 Zero
0x00e00000 +2.072680e-76 Normal
0xffda2177 -NAN
0x80000006 -4.941655e-83 Denormal
0x5f7473e5 +1.547922e+38 Normal
0x64702e63 +1.563578e+44 Normal
Unexpected EOF

```

TestFile8.bin

```

C:\UCSD Courses\Commo...
0x000eba1f +7.949073e-78 Denormal
0xffc00001 -NAN
0x7fffffff +NAN
0x014c0045 +6.563579e-76 Normal
0x7fc00000 +INF
0xffbffffff -1.157921e+77 Normal
0x80400000 -3.454467e-77 Normal
0x00400000 +3.454467e-77 Normal
0x80000001 -8.236092e-84 Denormal
0x003ffffff +3.454467e-77 Denormal
0x0000000b +9.059701e-83 Denormal
0x7fc00000 +INF
0x00000000 +0.000000e+00 Zero
0x80000000 -0.000000e+00 Zero
0x7fbffffff +1.157921e+77 Normal
0x00400000 +3.454467e-77 Normal
0x00000000 +0.000000e+00 Zero
0x7fe02ac6 +NAN
0x00000000 +0.000000e+00 Zero
0x00000000 +0.000000e+00 Zero
0xd0000000 -3.689349e+19 Normal
0x0e000000 +1.244603e-60 Normal
0x64702e63 +1.563578e+44 Normal
0x21cd09b4 +9.055762e-37 Normal
0x44206e69 +1.974913e+05 Normal
0x6d20534f +4.613071e+54 Normal
0xb40a0d0a -8.221305e-15 Normal
0x00000000 +0.000000e+00 Zero
0x00003a54 +1.229813e-79 Denormal
0xba037d00 -1.257067e-07 Normal
0x54623958 +7.421596e+24 Normal
0x5462390b +7.421507e+24 Normal
0xffdabcd8 -NAN
0xffc00000 -INF
0x5462380b +7.421212e+24 Normal
0x6369520b +9.174708e+42 Normal
0x54623968 +7.421614e+24 Normal
0x50000000 +3.689349e+19 Normal
0x2e65646f +1.341916e-21 Normal
0x0e7f0005 +4.939521e-60 Normal
0x80000000 -0.000000e+00 Zero
0x00e00000 +2.072680e-76 Normal
0xffda2177 -NAN
0x80000006 -4.941655e-83 Denormal
0x5f7473e5 +1.547922e+38 Normal
0xffc00000 -INF

```