UNCA CSCI 235
Final Exam Fall 2017
3:00-5:30 pm, Tuesday, 12 December, 2017

This is a closed book and closed notes exam. Communication with anyone other than the instructor is not allowed during the exam. **Furthermore, calculators, cell phones, and any other electronic or communication devices may not be used during this exam. Anyone needing a break during the exam must leave their exam with the instructor.** Cell phones or computers may not be used during breaks.

If you want partial credit for imperfect answers, explain the reason for your answer!

Name:_____

Problem 1 (15 points) C expressions
In the left column, there are twelve C expressions. Write their values in the right column. Express your answers in base 10. You may assume that all of these numbers are stored in 16-bit two's complement representation.

`~21`	`-22`
`!21`	`0`
`21 & 15`	`5`
`21 | 15`	`31`
`21 ^ 15`	`26`
`21 / 10`	`2`
`21 > 10`	`1`
`21 << 2`	`84`
`21 >> 2`	`5`
`0x10 + 10`	`26`
`2017 > 2016 + 1`	`0`
`15*235 && 0`	`0`

Problem 2 (6 points) Decimal to two's complement conversion
Convert the following four signed decimal numbers into **six**-bit *two's complement* representation. Some of these numbers may be outside the range of representation for **six**-bit two's complement numbers. Write "out-of-range" for those cases.

10 **001010**	**32** **out of range**
-10 **110110**	**-32** **100000**

Problem 3 (6 points) Decimal to Q4.4 conversion
Convert the following two signed decimal numbers into Q4.4 *two's complement* numbers (four fixed and four fractional bits) into signed decimal representation. If you can't express the number exactly, give the nearest Q4.4 representation.

-2.25 **11011100** or **-8 + 4 + 1 + ½ + ¼**
5.0 **01010000** or **4 + 1**
0.555 **00001001** or **0.5625** is closest

Problem 4 (6 points) Q4.4 to decimal conversion

Convert the following three Q4.4 *two's complement* numbers (four fixed and four fractional bits) into signed decimal representation.

01110000 **7.0** or **4 + 2 + 1**
11010100 **-2.75** or **-8 + 4 + 1 + ¼**
00000101 **0.3125** or **¼ + 1/16**

Problem 5 (6 points) Number puzzle

According to gdb, the C debugger, `(1/2017)*2017` is not 1. Why?

Because 1 and 2017 are integers, the result of dividing 1 by 2017 is the integer 0. So, (1/2017)*2017 is 0.

According to gdb, the C debugger, `(1/2017.0)*2017.0` is not `1.0` . Why?

Because 2017.0 is a float, the division is made using IEEE floating point arithmetic. However, due the precession of IEEE arithmetic, the result of (1.0/2017.0)*2017.0 is only accurate to about 7 decimal places, so it is unlikely to be exactly 1.0

Problem 6 (8 points) Printing a table

Write a loop, in C, to neatly print a table of squares from 0 to 99. Here are a few lines from the middle of the printout illustrating a possible output format.

```
30 * 30 =   900
31 * 31 =   961
32 * 32 =  1024
```

Write your code in the lines below.

```
for (int i=0; i<100; i++) {
  printf("%2d * %2d = %4d\n", i, i, i*i) ;
}
```

or something similar... 3 points for loop, 5 points for print

Problem 7 (15 points) Summing initial numbers

Suppose you have a file named “numbers.txt” containing lines which start with an integer and are followed by a single word. Here's an example of a four-line file that satisfies this requirement.

```
    1 lonely
   12   dozen
 2017 year
           13 unlucky
```

Write a few lines of C that:

- Open the file “`numbers.txt`” for reading.
- If “`numbers.txt`” cannot be read, print an error message and then exit.
- Read the lines of the file. You may assume that **all** lines are in the correct format: No error checking is expected!
- Print a neatly formatted listing of the number and the word for each input line. You may assume that no number is greater than 9999.
- After the last line is read, print the sum of all the numbers.

Here is an example of the program's output for the input shown above:

```
    1  lonely
   12  dozen
 2017  year
   13  unlucky
THE TOTAL OF THE NUMBERS IS 2043
```

Write your answer on the following page. It really doesn't require near that much room.

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

int main(int argc, char **argv) {

  // declarations
  char inWord[1000] ; // must be limit....
  int inNum ;
  int sumNum = 0 ;

  // 5 points in start
  // opening  – 2 points
  FILE *inFile = fopen("numbers.txt", "r") ;
  // checking – 1.5 points
  if (inFile == 0) {
    // error message and return – 1.5 points
    fprintf(stderr, "Can't open numbers.txt\n") ;
    return 1 ;
  }

  // 8 points in middle
  // read and scan loop  – 4 points
  while (fscanf(inFile, "%d %s", &inNum, inWord)==2) {
    // summing  – 1.5 points
    sumNum = sumNum + inNum ;
    // print  – 2.5 points
    printf("%4d  %s\n", inNum, inWord) ;
  }

  // 2 points in finish
  // final print – 2 points
  printf("THE TOTAL OF THE NUMBERS IS %d\n", sumNum) ;
  // following not required
  fclose(inFile) ;

   return (EXIT_SUCCESS) ;
}
```

Problem 8 (6 points) Which is faster?

Assume that X is declared as a 300 by 300 array of integers as follows:

```
int X[300][300] ;
```

Which of the following loops are likely to be executed the fastest on a modern computer. *Justify your answer!*

```
for (int i = 0; i<256; ++i)
     S = S + X[235][i] ;
```

```
for (int i = 0; i<256; ++i)
     S = S + X[i][235] ;
```

It all has to do with caching. If you access data that is close to recently accessed data, it is likely to be in a cache. In the loop on the left, the program is accessing memory sequentially as it goes through the loop iteration. In the loop on the right, memory accesses are 1024 (256*4) byte apart between In the loop in the left, the "stride" between loop iteration.

This what the picture on the cover of the textbook is illustrating.

Problem 9 (6 points) Structure and fields

Suppose you are given the following definition.

```
struct collegeInfo {
   char name[51] ;
   char city[21] ;
   int enrollment ;
}
```

Write a few lines of C code that

- Declare a variable wwc of type `struct collegeInfo`.
- Initialize wwc appropriately for a college named Warren Wilson College that is located in Swannanoa and has 753 students.

```
struct collegeInfo wwc ;
strcpy(wwc.name, "Warren Wilson College") ;
strcpy(wwc.city, "Swannonoa") ;
wwc.enrollment = 753 ;
```

or

```
struct collegeInfo wwc = {
     "Warren Wilson College",
     "Swannonoa",
     753
} ;
```

the following are *not* legal C

```
wwc.name = "Warren Wilson College" ;
wwc.city = "Swannonoa" ;
```

Problem 10 (6 points) On the average

You must show your work to receive full credit for this problem! Work the problem to completion!

Suppose a computer can retrieve a value from its cache in 20 µsec and from DRAM in 5 msec.

What is the average memory access time if the cache hit rate is 0.9?

0.9 * 20 µsec + 0.1 * 5000 µsec → 518 µsec

What is the average memory access time if the cache hit rate is 0.5?

0.5 * 20 µsec + 0.5 * 5000 µsec → 2510 µsec

Problem 11 (8 points) Cache structures

Suppose a puny computer with 16-bit address has a 4-way set-associate 2k (2048) byte cache and each block of the cache is 32 bytes:

- How many blocks does this puny cache have?

Cache-size/block-size → 2048/32 → 64

- How many sets does this puny cache have?

#blocks/#sets → 64/4 → 16

- The 16-bit address is divided into three fields: tag, index, and offset. How many bits are each allocated in each of these fields? (I suggest illustrating your point.)

offset: bits required to address block (32) → 5
index: bits required to address set (16) → 4
tag: what's left over → 16-5-4 → 7

Problem 12 (12 points) Definitions

To finish off, give short definitions of the following concepts, functions, hacks types, variables, etc., we have seen in this course.

Fell free to skip two: I will grade the best six of eight definitions.

`argv[2]` **second command line argument**
`digitalRead()` **Arduino function to read pin**
`FILE *` **C type used for accessing opened file**
GNU toolchain **Collection of compilers, linkers, etc., for C, C++, ...** **(by the way, it is not restricted to Linux)**
`pinMode()` **Arduino function to set pin mode (input or output)**
Randal E. Bryant **textbook author**
Raspberry Pi **Credit card size computer that can run Linux**
`strncmp()` **C routine to compare first n characters of string**