CMSC 313 Fall2009
Midterm Exam 1
Section 01
October 12, 2009

Name_____ Score _____

UMBC Username _____

Notes:

a. Please write clearly. Unreadable answers receive no credit.
b. For short answer questions your answer should be short, clear and to the point, not long and rambling.
c. For True / False question, write the word TRUE or FALSE; do not use the letters T and F. ***Using T and F will result in a 2-point deduction***.
d. There are no intentional syntax errors in any code provided with this exam. If you think you see an error that would affect your answer, please bring it to my attention.
e. You may assume that any necessary .h files have been #included where necessary.

True / False (2 points each)

1. _____ When an array is passed to a function a copy of the array elements is made for the function to manipulate.

2. _____ The **`sizeof( )`** operator is used to determine the number of bytes required to store a program object.

3. _____ The compiler will detect any attempt to use an invalid array index.

4. _____ If **`p`** is a pointer, then the expression **`p + 1`** adds the size of **`p`**'s type to **`p`**.

5. _____ For any two's complement integer, **`-X = ~X + 1`**

6. _____ Both **`scanf(`**) **`and fscanf(`**) can be used to read user input from the standard input.

7. _____ The size of a **`struct`** is at least as big as the sum of the sizes of its members.

8. _____ In C, functions are uniquely identified by their name, so no two .c files may have functions with the same name.

9. _____ The string **"Frodo LIVES!"** requires 12 bytes of memory.

10. _____ In C, all functions have global scope.

Short Answer

11. **(4 points)** Write two C statements that assign the value **`42`** to the **`56th`** element of an array of integers named **`ages`**.

    a. _____

    b. _____

12. **(4 points)** If not used carefully, C library functions for manipulating strings such as **`strcpy( )`** and **`strccat( )`** can lead to subtle errors or program termination (segmentation fault). Explain why this is so.

    _____

    _____

    _____

13. **(5 points)** Give a brief description of what this function accomplishes on the lines provided. DO NOT describe each line of code.

```
void mystery( double *dp, int n)
{
   int k, N = n - 1;

   for (k = 0; k < n / 2; --N, ++k)
   {
        double temp = *(dp + N);
         *(dp + N) = *(dp + k);
         *(dp + k) = temp;
   }
}
```

    _____

    _____

14. **(4 points)** Examine the code in the boxes, then answer the questions below

```
/** main.c **/

extern int randomInt;

static void printRandoms( int n)
{
  int k;
  for (k = 0; k < n; k++)
  {
     getRandomInt( 12345 );
     printf("%d\n", randomINt);
  }
}

int main( )
{
  printRandoms( 5 );
  return 0;
}
```

```
/** random.c **/

int randomInt;

void getRandomInt( int max )
{
   static long lastRandom = 100001;

  lastRandom
     = (lastRandom * 125) % 2796203;

  randomInt = (lastRandom % max) + 1;
}
```

a. What is the scope of `randomInt`?

b. What is the lifetime of `randomInt`?

c. What is the lifetime of `lastRandom`?

d. What is the scope of `lastRandom`?

15. **(30 points - 2 points for each blank)** Events at a gymnastics meet (high bar, rings, floor exercise, etc) are scored from 0.0 to 10.0 by each of 10 judges. The gymnast's score for the event is calculated by removing the highest and lowest of the judge's scores and then averaging the remaining 8 scores. Fill in the blanks for the functions below that calculate a gymnast's score for an event.

```
#define NR_SCORES 10


/* Calculates and returns a gymnast's score for one event
** from an array of scores provided by the user
** This function calls helper function SumScores( ) and
** GetHiLowScores( ) which are found on the next page */

_________ Gymnastics ( float _____________________________)

{
  float eventScore, hiScore, lowScore;

  eventScore = SumScores( scores );


  GetHiLowScores(scores, _________________, ____________________);


  eventScore -= ____________________________________;


  eventScore -= ____________________________________;


  eventScore /= ____________________________________;


  return eventScore;
}
```

```
/* Continuation from previous page
** Sums the values of an array of gymnasts scores */

_____________ SumScores( float _____________ )

{
      float sum = 0;
      int s;

      for (s = 0; s < NR_SCORES; s++)

        sum += _____________________;

      return sum;
}

/* "returns" the highest and lowest score via paremeters from an array
** of gymnastic scores which are known to be in the range 0.0 - 10.0 */

_______ GetHiLowScores (float scores[ ], __________hiScore, ________lowScore)

{
      float hi = -1.0, low = 11.0;
      int s;

      for (s = 0; s < NR_SCORES; s++)

      {
        if ( scores[s] > hi )

                hi = scores[s];

        if (scores[s] < low )

                low = scores[s];
      }

      _________________ = hi;

      _________________ = low;

}
```

16. **(6 points)** Convert the following values as indicated

    a. Binary 0110 0101 to hex _____

    b. Hex 0x2B to decimal _____

    c. Binary 0011 0110 to decimal _____

17. **(4 points)** Perform two's complement addition on each pair of 8-bit binary numbers. In each case, indicate whether overflow has occurred.

    1101 0011 + 0110 0001 = _____ Overflow? _____

    0110 0110 + 1000 0011 = _____ Overflow? _____

18. **(8 points)** Consider a 5-bit two's complement representation. Fill in the empty boxes in the following table. Addition and subtraction should be performed based on the rules for 5-bit, two's complement arithmetic

Number	Decimal Representation	Bit Representation
N / A	**-2**	
N / A	**9**	
N / A		**1 0011**
N / A		**0 1100**
TMax		
TMin		

19. **(10 points)** Assuming **8-bit integers** using two's complement representation, let **`int A`** have the value represented by the bit pattern 0110 1100, and **`unsigned int B`** have the value represented by the bit pattern 1000 0110. Fill in the empty boxes in the table below.

Expression	Value
(A & 0xF0) >> 2	(hex)
A | 0x3	(hex)
((A ^ B) << 3) && 0x7	(decimal)
~B | B	(hex)
(A & B) >> 2	(hex)

20. **(5 points)** Assuming 32-bit integers using two's complement representation, and given the declarations on the left, determine whether each statement in the table is ***always*** true. Circle YES if the statement is always true, circle NO if the statement is not always true.

```
int x = foo( );
int y = bar( );

unsigned int ux = x;
unsigned in uy = y;
```

Statement	Always True?	
x > 0 => x * 2 > 0	YES	NO
uy >> 3 == uy / 8	YES	NO
ux >= 0	YES	NO
x & 7 == 7 $\Rightarrow$ (x<<30) < 0	YES	NO
x > 0 && y > 0 => x + y > 0	YES	NO