ENEE 140, Fall 2016
Midterm Exam — Answer Key
Do Not Make a Copy!!

Date:

University of Maryland Honor Pledge: The University is committed to Academic Integrity, and has a nationally recognized Honor Code, administered by the Student Honor Council. In an effort to affirm a community of trust, the Student Honor Council proposed and the University Senate approved an Honor Pledge. The University of Maryland Honor Pledge Reads:

"*I pledge on my honor that I have not given or received any unauthorized assistance on this examination (or assignment)*"

Please write the exact wording of the Pledge, followed by your signature, in the space below:

Pledge: _____
Pledge: _____
Pledge: _____
Pledge: _____

Your signature: _____
Full name: _____ Course: _____ Directory ID: _____

List of Exam Questions:

Question:	1	2	3	4	5	6	7	Total
Points:	16	8	10	16	20	15	15	100
Score:								

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name, your section and your Directory ID on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.

- The exam has a maximum score of 100 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You may use any books or notes you like. Calculators are allowed, but no other electronic devices. Good luck!

1. (16 points) This problem tests your understanding of C types and casts and of C operators. Assume that variables `a`, `b`, `c` and `d` are defined as follows:

```
int      a = 10;
unsigned b = 9;
float    c = 8;
float    d = 7;
```

Fill in all the empty cells in the table below. For each of the C assignment expressions in the left column, state the resulting value of the `r2`–`r9` variables. If an expression results in a compilation error, write ERROR.

Assignment		Value
float	r1 = c / a;	0.8
unsigned	r2 = b / a;	**0**
unsigned	r3 = b % 2;	**1**
unsigned	r4 = c % 2;	**ERROR**
unsigned	r5 = INT_MAX % 2;	**1**
float	r6 = (**float**)(b/a) * 10.0;	**0**
float	r7 = c / a * (**int**)(d+3);	**8**
int	r8 = a++ + ++b;	**20**
float	r9 = c // *d;*	**ERROR**

2. (8 points) This problem tests your understanding of integer arithmetic. Choose one of the following answers. Assume operations take place on the grace machine (w = 32).

UINT_MAX − 2∗INT_MAX =

A. 0

B. 1

C. INT_MIN

D. $2^{31} - 1 = 2147483647$

2. **B**

Solution:

$$\begin{aligned} \text{UINT\_MAX} - 2{*}\text{INT\_MAX} &= 2^{32} - 1 - 2 * (2^{(32-1)} - 1) \\ &= 2^{32} - 1 - 2^{32} + 2 \\ &= -1 + 2 \\ &= 1 \end{aligned}$$

3. (10 points) This problem tests your understanding of loops. Convert the following `while` loop into a `for` loop.

```
int x = -10;
int y = 0;
while (y <= 500) {
    printf("%d %d\n", x, y);
    x += 5;
    y += x;
}
```

Solution:

```
int x = -10;
int y = 0;
for (y = 0 ; y <= 500; y += x) {
    printf("%d %d\n", x, y);
    x += 5;    // y must be incremented after x!
}
```

4. (16 points) This problem tests your understanding of characters and their numerical representation (ASCII values). The function `avg()` is supposed to read the user input character-by-character until EOF is reached, and to return the average ASCII value of the input characters as a float. (The average = the sum of a list of numbers divided by the number of numbers). Fill in the blanks below to complete the implementation.

EXAMPLE: for the input: `abc` the return value should be 98.000000

```
    float     avg (){

float avg(){

  int sum      = 0      ;
  int c;
  int i       = 0       ;
  c = getchar();
  while(   c != EOF   ) {
     sum = sum + c;
        i++;
     c =   getchar();
  }

  return  (float)sum / i ;
}
```

Solution:

```
float avg(){
  int sum = 0;
  int c;
  int i = 0;
  c = getchar();
  while ( c != EOF ){
    sum = sum + c;  // or sum += c;
    i++; // or i = i + 1; or i += 1;
    c = getchar();
}

  return (float)sum / i;  // or sum/(float)i or (float)sum/(float)i;
}
```

5. (20 points) This problem tests your understanding of functions and arithmetic operations.

   The number of ways to choose k items out n total items (when replacement and order don't matter) is given by the formula: $\frac{n!}{k!*(n-k)!}$, where $n! = n*(n-1)*(n-2)*...*2*1$.

   For example, choosing $k = 2$ options out of $n = 5$ gives you $\frac{5!}{2!*(5-2)!} = \frac{5*4*3*2*1}{2*1*3*2*1} = 10$.

   Implement the functions `int factorial(int a)` and `int choose(int n, int k)` where the function `int factorial(int a)` returns the $a!$ and the function `int choose(int n, int k)` returns the result of $\frac{n!}{k!*(n-k)!}$.

```
#include <stdio.h>

int factorial(int a);
int choose(int n, int k);

int main(void)
{
        int n, k;
        printf("Enter in n, followed by k\n");
        scanf("%d", &n);
        scanf("%d", &k);
        printf("number of possibilities = %d\n", choose(n, k));
        return 0;
}

int factorial(int a)
{

}
```

```
int choose(int n, int k)
{

}
```

Solution:

```
int factorial(int a)
{
        int i = 1;
        while(a > 0)
        {
                i *= a--;
        }
        return i;
}

int choose(int n, int k)
{
        return factorial(n)/(factorial(k)*factorial(n-k));
}
```

6. (15 points) This problem tests your understanding of loops and bitwise operations. Write the output of the following program.

```
#include <stdio.h>

int main(void)
{
    int count   = 0;
    int max     = 10;
    int i       = 1;
    int c       = 2;

    printf("count\tmax\ti\tc\t\n");

    for(count = 1; count < max; count++)
    {
        if( count++ == 3 )
            c |= i;
        printf("%d\t%d\t%d\t%d\n", count, --max, i, c);

    }
```

}

Solution:

count	max	i	c
2	9	1	2
4	8	1	3
6	7	1	3

7. (15 points) This problem tests your understanding of command line arguments and strings. What is the output of this program when the compiled as `a.out` and invoked from the command line as follows:

```
a.out Good Luck On The Final!
```

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  int x;
  printf("argc=%d\n", argc);
  for(x=0; x<argc; x++)
    argv[x][x % 3] = '\0';
    printf("argv[%d]=\"%s\"\n", x, argv[x]);
}
```

Solution:

```
argc=6
argv[0]=""
argv[1]="G"
argv[2]="Lu"
argv[3]=""
argv[4]="T"
argv[5]="Fi"
```