ENEE 140, Spring 2023
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	11	12	15	15	15	16	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 name and section at the bottom of each page as well.

- 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. No electronic devices (e.g. laptops, tablets, smartphones, calculators) are allowed. 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:

```
unsigned    a = 2;
float       b = 1;
int         c = −1;
double      d = −2;
```

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 `r1`–`r8` variables. If an expression results in a compilation error, write ERROR.

Assignment		**Value**
int	r0 = a / 2;	1
float	r1 = b / 2;	**0.5**
float	r2 = c / 2;	**0.0**
int	r3 = 2 ∗ c++;	**-2**
char	r4 = '0' + a;	**'2'**
unsigned	r5 = a % 3;	**2**
unsigned	r6 = a % a;	**0**
unsigned	r7 = UINT_MAX + a;	**1**
int	r8 = ++a / d;	**-1**

2. (11 points) This problem tests your understanding of function prototypes. Suppose we want a function named average that takes an array of integers and its size as an integer and returns the average value of the array as a float. Which of the following is the correct function prototype?

   A. **int** average(**float** arr [], **int** size);

   B. **float** average(**int** arr [], **int** size);

   C. **float** average(**int** arr, **int** size);

   D. **int** average(**int** arr, **int** size);

2. **B**

Solution:

B: float average(int arr[], int size);

3. (12 points) This problem tests your understanding of increment/decrement operators. What is the correct output of the following program?

```
#include <stdio.h>

int main()
{
    int a=3;
    int b=5;
    int c=2;

    b = 2 * ++a;
    c += b++;
    c %= 4;

    printf("a=%d b=%d c=%d", a, b ,c);

}
```

   A. a=4 b=7 c=3

   B. a=3 b=9 c=5

   C. a=4 b=9 c=2

   D. a=3 b=9 c=3

3. **C**

Solution:

First line is a prefix increment operator. This means a will be 4 and b will also be 4 after this line.

Second link is postfix increment operator so 4, the current value of b, is added to c making c 10. b then is incremented to 9.

The line c %= 2 is equivalent to c = c % 2. The remainder when dividing 10 by 4 is 2.

The final values are then C: 4, 9, and 2.

4. (15 points) This problem will test your understanding of integer and floating point arithmetic, arrays and loops.

Jahmir Young has been lighting it up on the court for the University of Maryland's men's basketball team. The program on next page is meant to calculate his points per game for the last five games and print it out to `stdout` accurate to 2 decimal places.

Fill in the blanks within `main()` to make this occur.

```
#include <stdio.h>

int main()
{
    int points[5] = {9,18,11,16,20};
    ____float____ points_per_game;
    int total=0;

    for(int i = 0; i < 5 or i <= 4; i++){
        total += points[____i____];
    }

    points_per_game = (float) or 1.0* total / 5;

    printf("Points␣per␣game:␣____%.2f____", points_per_game);

    return 0;
}
```

Solution:

```
#include <stdio.h>

int main()
{
    int points[5] = {9,18,11,16,20};
    float points_per_game;
    int total=0;

    for(int i = 0; i < 5 or i <= 4; i++){
        total += points[i];
    }

    points_per_game = (float) or 1.0* total / 5;

    printf("Points␣per␣game:␣%.2f", points_per_game);

    return 0;
}
```

5. (15 points) This question will test your ability to understand and debug code. Complex numbers written as $a + bj$, where a and b are real numbers, can be converted to exponential form in the following way: $Ae^{\phi j}$, where $A = \sqrt{a^2 + b^2}$, $\phi = atan(\frac{b}{a})$. Your friend has written a function that does this conversion and prints the complex number in exponential form, but they have not taken ENEE140 and so their code is riddled with errors. There are five errors on five lines. Find all five errors, and in the space bellow the function write the line number of each error along with a correction. There are no errors with the method of conversion, only its implementation into C.

```
 1
 2 #include <math.h>
 3 #include <stdio.h>
 4
 5 void convert_to_exp_form(int re, int im) {
 6
 7     float amp = sqrt(re^2 + im^2); //math.h function for the square root
 8     float phase = atan(im/re); //math.h function for the inverse tan
 9
10     if(im > 0 && re < 0) //adjust phase to the correct quadrant
11         phase += M_PI; //math.h pi constant
12     else if(im <= 0 && re < 0)
13         phase += M_PI;
14     else (im < 0 && re >= 0)
15         phase += 2*M_PI;
16
17     printf("The␣number␣%.2d␣+␣%.2dj␣is␣%.2de^(%.2dj)\n", re, im, amp, phase);
18
19     return 0;
20 }
```

Solution:

```
Line 5: void convert_to_exp_form(float re, float im) {
Line 7: float amp = sqrt(re*re + im*im);
Line 14: else if(im < 0 && re >= 0)
Line 17: printf("The␣number␣%.f␣+␣%.2fj␣is␣%.2fe^(%.2fj)\n",
re, im, amp, phase);
Line 19: just return, or nothing
```

6. (15 points) This question will test your understanding of input/output, character data types, and logical statements. Determine the output of the program below with the input:

   Umd 4EVER!

```
#include <stdio.h>

int main() {

    int c;

    c = getchar();
    while(c != EOF) {
        if((c >= 'a') && (c <= 'z'))
            putchar('#');
        else if((c >= 'A') && (c <= 'Z'))
            putchar(c + ('a' - 'A'));
        else
            putchar('$');

        c = getchar();
    }
    return 0;
}
```

Solution:

```
u##$$ever$
```

7. (16 points) This question will test your understanding of loops, functions, control flow, and mathematical/logical operations in C. Determine the output to the following program:

```
#include <stdio.h>

int func1(int a) {
    int sum = 0;
    int tracker = 10;
    for(int i = 0; i <= a; i++) {
        sum += i/a;
    }
    return sum;
}

int func2(int tracker, int a) {
    tracker = tracker|7;
    if(tracker >= 10)
        tracker *= a;
    return tracker;
}


int main() {
    int tracker = 5;

    tracker = func1(tracker);
    printf("Tracker is %d\n", tracker);

    for(int i = 0; i < 3; i++) {
        tracker += ( (i % 2) == 0);
    }
    printf("Tracker is %d\n", tracker);

    tracker = tracker*4 % 10;
    printf("Tracker is %d\n", tracker++);

    func2(tracker, 4);
    printf("Tracker is %d\n", tracker);

    return 0;
}
```

Solution:

```
Tracker is 1
Tracker is 3
Tracker is 2
Tracker is 3
```