ENEE 140, Fall 2024
Midterm Exam

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	8	Total
Points:	16	10	12	20	12	12	18	12	112
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 last question is for EXTRA CREDIT and is worth 12 points. You may receive full credit without answering it, assuming that you answered correctly all the other questions (the exam will be scored out 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 program tests your understanding of for loops and arrays. Create a loop that squares the odd values in this string only for the second half of the array (hint - the "second half" starts with index 5 which contains the value 6 for this array).

```
#include <stdio.h>

int main() {
    int arr[] = {8,2,5,4,24,6,7,17,9,2};
    int i;

    for (i = _____; i < 10; i++) {
        if (i > _____ && (arr[i] % _____) == _____) {
            arr[i] = arr[i]*arr[i];
            }
        }
    }
    return 0;
}
```

2. (10 points) This question tests your understanding of characters in C. Examine the function below.

```
char fun1 (char c) {
    if (c >= 'A' && c <= 'Z') {
        c = 'a' + (c - 'A');
    }
    else if (c >= 'a' && c<= 'z') {
        c = 'A' + (c - 'a');
    }

    return c;
}
```

This is a two-part question.

(a) What does this function do? Include descriptions of both input and output.

(b) Write a one line statement to print the output of the function when c = 'J'.

3. (12 points) This question tests your understanding of random number generation. Write one-line expressions using `rand()` to generate the given sequences:

a) Integers from 0 to 51 (inclusive)

b) Integers from 100 up to but not including 200

c) Multiples of 5 between 10 and 100 (inclusive)

4. (20 points) This question tests your ability to understand format specifiers and type conversions. In the blanks, fill in the output of each printf function.

   In order to show that a space is printed, please just put a "/"

   Assume that these functions are lines in a working program with the following variables:

```
int i = 4;
float j = 3.289;
unsigned k = UINT_MAX-2;
```

   (a) printf("%4d", k);

   (a) _____

   (b) printf("%02u", i);

   (b) _____

   (c) printf("%3d", -i);

   (c) _____

   (d) printf("%3.1f", j);

   (d) _____

   (e) printf("%3.2d", i);

   (e) _____

5. (12 points) The following three questions (which continue onto the **next page**) will test your ability to understand binary-decimal conversions, bitwise operators, and bit importance. Only one of the options is correct for each question. Indicate which is on the given **line** below. A table of bitwise operators is provided below for your reference.

X	Y	X&Y	X|Y	X^Y	~(X)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

(a) The following two numbers on the left-hand side below are in **base 2** (binary form). A bitwise operator is applied to them and the result is given in **base 10** (decimal form) on the right-hand side. The bases are indicated in the subscripts of each number for reference. Identify which operator was used to obtain this result.

$(01110)_2$ _____ $(11010)_2 = (20)_{10}$

A. ^

B. &

C. |

D. ~

(a) _____

(b) Assume that you have three variables, `a`, `b`, and `c`, and `result` defined as follows:

```
int a = 2;
float b = 1, c = 3;
float result = (int)((UINT_MAX % a) + b) / (int)(c + a);
```

What is the value of `result`? If needed, assume operations take place on the grace machine (w = 32).

A. 2

B. 1

C. 0.4

D. 0.0

E. UINT_MAX

(b) _____

(c) An **unsigned** number's first 8 bits can be represented in the following expression, where b_7 is the leftmost bit and b_0 is the rightmost bit:

$b_7b_6b_5b_4b_3b_2b_1b_0$

If one of these eight bits were to be flipped (have their value switched from 0 to 1 or vice versa), which one would have a bigger impact on the overall **value** of the number **and** why?

A. b_0 because it represents the least significant bit and flipping it has the most direct effect on small increments.

B. b_7 because it represents the largest power of 2, contributing the most to the total value.

C. b_7 because it represents the sign of the number, and flipping it will drastically change the value.

D. b_6 because flipping a bit this close to the most significant bit alters the upper half of the binary number, impacting the overall value significantly.

(c) _____

6. (12 points) This question will test your understanding of operators and loops.

   What is the output of the following code?

```
1  #include <stdio.h>
2  int main(){
3      int i = 5, k = 7;
4      int j = k++;
5      while(++i < 10){
6          if(i < j){
7              printf("moo, i:%d, j:%d\n", i, j);
8          }
9          else {
10             printf("oink, i:%d, j:%d\n", i, j);
11         }
12     }
13     return 0;
14 }
```

7. (18 points) This question will test your ability to debug code. The following program is designed to calculate the sum of an array of integers using a separate function called `calculateSum`. The function is intended to take an array and its `size` as arguments and return the total sum of the array's elements. However, there are errors (technical and functional) in the entirety of the code that prevent the program from working correctly. Circle the lines that contain bugs and correct them on the given lines/space below the program. (Hint: There are **six lines with errors** in total.)

```
1   include <stdio.h
2
3   int calculateSum(int arr[], int size);
4
5   int main() {
6       int array[5] = {100, -50, 200, 300, -150};
7       int result;
8
9       result = calculateSum(array);
10      printf("The sum of the array is: %f\n", result);
11      return 0;
12  }
13
14  int calculateSum(int arr[], int size) {
15      int sum = 0;
16      for (int i = 1; i <= size; i++) {
17          sum = arr[i];
18      }
19      return;
20  }
```

8. (12 points) EXTRA CREDIT

This problem tests your understanding of `if-else` statements. What is the output of the following function?

Note: the code continues on the **next page**.

```
#include <stdio.h>

int main() {
    int x = 1, y = 2, z = 3, w = 4, p = 5, q = 6;

    if (y % 2 == 0) {
        y = 4 * (x + 2);
        p = 3 * (z + 1);
    }

    if (x % 3 == 1) {
        y = 5 * (z + 3);
        z = 2;
        if (w % 3 == 0) {
            x = 6 * (y + 1);
            q = 3 * (p / 3);
        }
        w = 2 * (y + 4);
    } else {
        x = 8 * (w + 1);
        y = 7 * (z + 2);
    }

    if (y % 4 == 0) {
        if (z % 2 == 1) {
            w = 3 * (x + 2);
        } else {
            w = 4 * (y - 3);
        }
    } else {
        if (w % 2 == 0) {
            x = 5 * (z + 1);
        }
        z = 2 * (y + 5);
    }

    if (w % 5 == 0) {
        x = 7 * (y - 2);
        y = 3 * (z + 1);
        if (z % 3 == 0) {
            w = 2 * (x + 3);
```

```
42                  p = 4 * (y / 2);
43                  q = 5 * (z + 1);
44              }
45          } else {
46              p = 6 * (w + 1);
47              z = 3 * (x - 1);
48              x = 5 * (y + 2);
49          }
50
51          printf("x is %d, y is %d, z is %d, w is %d, p is %d, q is %d\n"
52          , x, y, z, w, p, q);
53          return 0;
54      }
```