ENEE 140, Spring 2024
Final 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	9	Total
Points:	16	8	6	8	6	15	23	18	10	110
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 10 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 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 = 255;
char         c = 'A';
float        d = 3.14;
```

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
float	r0 = d / 2;	1.57
int	r1 = b / 10;	
char	r2 = c + 'c' − 'C';	
float	r3 = d % (d / 2);	
unsigned	r4 = b % (a * a);	
float	r5 = (**int**) d + a;	
float	r6 = d − (**int**) d;	
unsigned	r7 = (a < ++a) ? a + 1 : a − 1;	
int	r8 = b == b++;	

2. (8 points) For each of the following **four** questions (which continue on the **next page**) only one answer is correct. Indicate which one on the given lines below.

   (a) If an integer "a" has a value of 5, what is the new value of "a" after the following statement:

   ```
   a = 2*((a & 3) | 2)
   ```

   A. 0

   B. 2

   C. 6

   D. 8

   (a) _____

   (b) Suppose you are given a compiled program called **"testudo_loveletter"**. You're told to run this program using the following command line text:

   ```
   ./testudo_loveletter terp 21 secret.txt
   ```

   With this information, what would `argc` and `argv[2]` be?

   A. argc equals 4 and argv[2] is "21"

   B. argc equals 4 and argv[2] is "terp"

   C. argc equals 4 and argv[2] is "secret.txt"

   D. argc equals 3 and argv[2] is "21"

   E. argc equals 3 and argv[2] is "terp"

   (b) _____

   (c) The following assignment (assume `h` is of type `int`) results in a compilation error in C. Why is this the case?

   ```
   h = 2 % (double)h;
   ```

   A. The % operator can't be used on a variable that is being reassigned.

   B. h was not declared as a const int.

   C. The value of h wasn't initialized.

   D. The % operator can't be used on a double.

   (c) _____

(d) If an integer `d` has a value of 7, what will its new value be after the following statement:

d = 3 + ++d >> 1

A. 4

B. 6

C. 7

D. 5

(d) _____

3. (6 points) These multiple choice questions test your understanding of functions. Circle your answer(s) and put them in the corresponding lines below. Note that each question may have **multiple correct answers**.

(a) Which of the following is true about the **return type** of functions in C?

A. Functions can return any type

B. Functions can return any type except array and strings

C. Functions can return any type except array, strings and union

D. Functions can return nothing, because the return type must be **void**

(a) _____

(b) In C, what is a function **definition**?

A. A prototype that specifies a function's name, return type, and parameters

B. A prototype that specifies a function's name and return type

C. A block of code that specifies a function's behavior

D. A variable declared within a function

(b) _____

(c) What is the purpose of a function prototype in C?

A. To define the return type of the function

B. To declare the function before it is called

C. To provide a brief description of the function's functionality

D. To specify the number and type of parameters a function expects

E. To identify the data types functions can handle

(c) _____

4. (8 points) This problem tests your understanding of loops and number base conversion. Consider the following C code snippet that accepts a decimal input and uses a `while` loop to print this number in octal (base 8):

```
1  #include <stdio.h>
2
3  int main() {
4      int decimal, octal_digit = 0;
5      printf("Enter a decimal number: ");
6      scanf("%d", &decimal);
7
8      while (decimal != 0) {
9          octal_digit = decimal % 8;
10         printf("%d", octal_digit);
11         decimal = decimal/8;
12     }
13     return 0;
14 }
```

   (a) Convert the `while` loop in the above code snippet into a `for` loop.

   (b) b. There is a simpler way to do this, using a single `printf` (without loops). What should the printf statement be?

5. (6 points) This question will test your understanding of `switch` statements. Convert the code segment **on the next page** from using `if/else` statements to using `switch` statements. Your answer **should only** include code **pertaining to the segment below** and nothing else: **don't include** statements for header files, `main`, variable declarations, etc.

```
if (check == 0 || check == 1 || check == 2 || check == 8) {
    printf("The value of check is %d.\n", check);
}
else if (check == 4 || check == 5) {
    printf("The value of check is too weird!\n");
}
else if (check == 6) {
    printf("There is no need for this value.\n");
}
else {
    printf("Check is irrelevant.\n");
}
```

Continue the code below.

```
switch (check) {
```

6. (15 points) This problem will test your understanding of using C programs to solve mathematical problems alongside debugging.

   The following program takes a user-given input for two **integer points (in the xy plane)** of a line and stores them into a `point` structure data type. Then, it uses those two data points to find the x-intercepts, y-intercepts, slopes, and slope-intercept form equations of the that line and another line **perpendicular** to it (assume all these values **are finite, nonzero, and floating point**). The lines **intersect at the second given point**. Finally, it prints the equations of the lines, **adding two decimal places wherever appropriate**. Circle the lines that contain bugs and correct them on the given lines/space **on the next page**. (Hint: There are **5 lines with errors**).

   Remember the slope-intercept form equation of a line is $y = mx + b$, where m is the slope of the line and b is its y-intercept. Additionally, two lines are perpendicular when their slopes are negative reciprocals of each other. Finally, note that the errors can be both syntax or mathematical.

```
1  #include <stdio.h>
2
3  typedef struct point_type {
4      int x:
5      int y;
6  } point;
7
8  int main(){
9      int p1, p2;
10     float slope, slope_perp, x_int, x_int_perp, y_int, y_int_perp;
11
12     /* Enter Point 1 & Point 2 Info: (x, y) Order */
13     scanf("%d-%d", p1.y, p1.x);
14     scanf("%d-%d", &p2.x, &p2.y);
15
16     /* Find Slopes, Y-Intercepts, & X-Intercepts */
17     slope = (p2.y - p2.x)/(p1.y - p1.x);
18     slope_perp = -1/slope;
19
20     y_int = p2.y - (slope * p2.x);
21     y_int_perp = p2.y - (slope_perp * p2.x);
22
23     x_int = -y_int/slope;
24     x_int_perp = -y_int_perp/slope_perp;
25
26 /* Print Equation Of Regular Line Then Perpendicular One */
27 printf("y-=-%.2dx-+-%.2d.-X-Intercept:-%.2d.\n,-slope,-y_int,-x_int");
28 printf("y-=-%.2fx-+-%.2f.-X-Intercept:-%.2f.", slope_perp, y_int_perp, x_int_perp);
29
30     return 0;}
```

Name: _____ Page 7 Section: _____

7. (23 points) This question will test your ability to correctly open files and perform specific operations within them. The program **on the next page** takes the following command line arguments, not including the executable: "input.txt" and "output.txt". You may assume that argv[1] = "input.txt" and argv[2] = "output.txt" and that they always exist like this. The program should read all the lines of "input.txt" and put them in "output.txt" with the following restrictions:

   1. Every **odd** numbered line in "output.txt" should **only** include the **letters** (no numbers, whitespace, or any other type of special characters) of the corresponding odd line in "input.txt".

   2. Every **even** numbered line in "output.txt" should **only** include the **numbers** (no letters, whitespace, or any other type of special characters) of the corresponding even line in "input.txt".

   Below is an example for the input and output text files.

"input.txt":	"output.txt":
Hello World 1234!	HelloWorld
ENEE140 is the best!	140
Testing 1, 2, 3.	Testing
Clark School!!!!2024	2024

   Fill in the blanks to make the program works properly (located on the next page). Assume the first line is line 1.

```
#include <stdio.h>
#define MAX_STRING 1000

int main(int argc, char *argv[]) {
    _____ *input, *output;
    char line[MAX_STRING];
    int i, count = 1;

    /* open files */
    input = fopen(_____, _____);
    output = fopen(_____, _____);

    /* reading each line from file and writing to other file */
    for (; fgets(_____, _____, _____); count++) {
            i = 0;

            for (; line[i] != _____; i++) {
                /* odd vs. even lines */
                if ((_____ % _____) == 1) {

                    if ((line[i] _____ && _____ <= 'Z') ||
                    (_____ && _____)) {
                        fputc(_____, _____);
                        }
                }

                else {
                    if (line[i] >= _____ && line[i] <= _____) {
                        fputc(_____, _____);
                        }
                    }
                }

        /* add newline to output*/
        fputc(_____, _____);
    }
    return 0;
}
```

8. (18 points) This problem tests your understanding of `if-else` statements. What is the output of the following program? The code continues **on the next page**. (Note: There will be negative values, so don't panic)

```
1  #include <stdio.h>
2
3  int main() {
4      int a = 0, b = 0, c = 0, d = 0, e = 0, f = 0;
5
6      if (b % 2 == 1) {
7          b = 3 * (a + 2);
8          e = 5 * (c + 3);
9      }
10
11     if (a % 3 == 0) {
12         b = 2 * (c + 1);
13         c = 0;
14
15         if (d % 2 == 1) {
16             a = 5 * (b - 1);
17             f = 2 * (e / 2);
18         }
19
20         d = 4 * (b + 3);
21
22     } else {
23
24         a = 7 * (d + 2);
25         b = 9 * (c + 4);
26     }
27
28     if (b % 5 == 2) {
29         if(c % 2 == 1) {
30             d = 2 * (a + 3);
31         } else {
32             d = 3 * (b - 2);
33         }
34
35     } else {
36
37         if(d % 2 == 1) {
38             a = 4 * (c - 1);
39         }
40
41         c = 3 * (b + 2);
42     }
43
```

```
44      if (d % 4 == 0) {
45          a = 4 * (b - 3);
46          b = 6 * (c + 1);
47
48          if (c % 5 == 0) {
49              d = 7 * (a + 2);
50              e = 8 * (b / 4);
51              f = 9 * (c - 2);
52          }
53
54      } else {
55
56          e = 7 * (d + 1);
57          c = 6 * (a - 2);
58          a = 4 * (b + 3);
59      }
60
61      printf("a is %d, b is %d, c is %d, d is %d, e is %d, f is %d", a,b,c,d,e,f);
62
63      return 0;
64  }
```

9. (10 points) EXTRA CREDIT: This question tests your understanding of C variables and operations. The Clone Wars rage on, and now it's your job to determine the many values and forms CLONEWARS takes as it passes through the code. To do this, you should track the progress of CLONEWARS across the program, keep in mind all changes to its value, and write down what each of the printf statements will print out. The code continues **on the next page**. Don't get lost in the war, you must prevail!

```
1  #include <stdio.h>
2
3  int REPUBLIC (int x);
4  int SEPARATIST (int y);
5  void NEUTRAL (int z);
6
7  int main() {
8      int CLONEWARS = 303;
9      printf("CLONEWARS = %d\n", CLONEWARS);
10
11     CLONEWARS = REPUBLIC(CLONEWARS);
12     printf("CLONEWARS = %d\n", CLONEWARS);
13
14     CLONEWARS += ((10 & 7) / 2) | (8 >> 2);
15     printf("CLONEWARS = %d\n", CLONEWARS);
16
17     NEUTRAL(CLONEWARS);
18     printf("CLONEWARS = %d\n", CLONEWARS);
19
20     CLONEWARS += 0x2B;
21     printf("CLONEWARS = %d\n", CLONEWARS);
22
23     CLONEWARS = SEPARATIST(CLONEWARS);
24     printf("CLONEWARS = %d\n", CLONEWARS);
25
26     return 0;
27 }
28
29 int REPUBLIC (int x) {
30     int acclamator;
31     int arquitens = 0;
32     for (acclamator = 0; acclamator < 3; acclamator++) {
33         arquitens = acclamator*2-arquitens;
34     }
35     if (arquitens > 20) {
36         return x+2;
37     } else {
38         return x+1;
39     }
40 }
```

```
41
42 void NEUTRAL (int z) {
43     int mandalore = 0;
44     do {
45         mandalore += z+mandalore;
46         z = mandalore % 30;
47     } while (mandalore % z !=0);
48 }
49
50 int SEPARATIST (int y) {
51     int munificent;
52     int providence = 0;
53     for (munificent = 2; munificent > 0; munificent--) {
54         providence = (munificent*2);
55     }
56     if (providence > 4) {
57         y = 150;
58         return y;
59     } else {
60         y = 140;
61         return y;
62     }
63 }
```