ENEE 140, Fall 2025
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	8	9	Total
Points:	14	6	12	8	16	12	12	20	14	114
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 14 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. (14 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:

```
float a = 2.0;
int b = 6;
char c = 'd';
unsigned d = 10;
```

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`–`r8` variables. If an expression results in a compilation error, write ERROR. You may use what you like from `limits.h` (ex. `INT_MIN`)

Assignment		Value
float	`r1 = b/2.0`	3.0
unsigned	`r2 = UINT_MAX + 1;`	**0**
int	`r3 = b % a;`	**ERROR**
int	`r4 = b++;`	**6**
int	`r5 = b % 4 + 1;`	**3**
char	`r6 = ++c;`	**'e'**
float	`r7 = (float)(3/b);`	**0.00**
unsigned	`r8 = INT_MAX % 2;`	**1**

2. (6 points) This problem tests your understanding of integer overflow.

   Consider the following C program:

```c
#include <stdio.h>
#include <limits.h>

int main() {
    unsigned int u = UINT_MAX;
    signed int s = INT_MIN;
    int a, b;

    a = s + INT_MAX;
    b = u - 3;
    printf("a = %d\n", a);
    printf("b = %d\n", b);

    return 0;
}
```

   What does the program print out?

   Solution: The program should print out the following:

```
a = -1
b = -4
```

3. (12 points) This question will test your understanding of the difference between **pre-increment (++i)** and **post-increment (i++)**. Fill in the blanks with the appropriate output for each printf statement.

```c
#include <stdio.h>

int main() {
    int i = 5;

    printf("%d\n", i++);          // OUTPUT: 5
    printf("%d\n", (++i*i));      // OUTPUT: 49
    printf("%d\n", i++);          // OUTPUT: 7
    printf("%d\n", --i);          // OUTPUT: 7

    return 0;
}
```

Solution:

```
Output: 5
Output: 49
Output: 7
Output: 7
```

4. (8 points) This question tests your understanding of **function prototypes** in C.

   You are writing a program that will later define a function named `print_course_information`. The function:

   - returns **no value**,
   - takes in the following parameters:
     1. a student's UID number (`int`),
     2. a course grade (`float`), and
     3. a letter grade (`char`).

   Task: Write the correct **function prototype** for `print_course_information`.

   Solution:

   ```
   void print_course_information(int uid, float grade, char letter);
   ```

5. (16 points) This problem tests your understanding of printing in C. Write printf statements with the following outputs ('␣' represents a space):

   ```
   a. %%%
   b. __256
   c. "There are x students taking ENEE140.", where x is a variable storing the
   number of students
   d. The value of UINT_MIN
   ```

   Solution:

   ```
   a. printf("%c%c%c", '%', '%', '%'); printf(%% %% %%);
   b. printf("%5d", 256);
   c. printf("There are %d students taking ENEE140.", x); using %d or %u is fine
   d. printf("%u", 0); or printf("%u", UINT_MIN); (%d is fine)
   ```

6. (12 points) This question will test you on your **for loop** knowledge and understanding. We are looking to find the sum of all the numbers from 2 to 25 (inclusive), but only the **ODD** numbers. Make sure to iterate through every number between 2 and 25 to calculate the total of all the odd numbers.

```
int summation() {
    int sum = 0;
    for (int i = 2; i <= ____25____; i____++____) {
        if (i ____%2____ != 0) {
            sum ____+=____ i;
        }
    }
    return sum;
}
```

Solution:

```
int summation() {
    int sum = 0;
    for (int i = 2; i <= 25; i++) {
        if (i %2 != 0) {
            sum += i;
        }
    }
    return sum;
}
```

7. (12 points) This problem tests your understanding of the modulus operator and the getchar() function in C.

Consider the following C program:

```
#include <stdio.h>

int main() {
    int temp = 0;
    int c = 0;

    while (temp < 10) {
        c = getchar();
        if (temp % 2 == 1) {
            printf("%c", c);
        }
        ++temp;
    }
    return 0;
```

```
}
```

Write what the program prints out after a user inputs the following text:

ABCDEFGHIJKL

Solution: The program should print out the following:

BDFHJ

8. (20 points) This problem test your understanding of computer arithmetic and functions. Read through the program below and provide its output.

```
#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(void) {
    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 2
```

9. (14 points) (BONUS) This question tests your understanding of C variables and operations. Your goal is to track the variable known as impostor as it progresses through the code. Note the changes to its value, and write down what each of the `printf` statements will print out. Keep an eye on it, it's moving sus!

```
1
2  #include <stdio.h>
3
4  int cafeteria (int x);
5  int electrical (int y);
6  void shields (int z);
7
8  int main() {
9      int impostor = 140;
10     printf("Impostor value is: %d\n", impostor);
11
12     impostor = cafeteria(impostor);
13     printf("Impostor value is: %d\n", impostor);
14
15     impostor = impostor + 67 - 104;
16     printf("Impostor value is: %d\n", impostor);
17
18     shields(impostor);
19     printf("Impostor value is: %d\n", impostor);
20
21     impostor -= 1;
22     printf("Impostor value is: %d\n", impostor);
23
24     impostor = electrical(impostor);
25     printf("Impostor value is: %d\n", impostor);
26     printf("The Impostor is: %c\n", (char)impostor);
27
28     return 0;
29 }
30
31 int cafeteria (int x) {
32     int vent;
33     int kill = 0;
34     for (vent = 0; vent < 3; vent++) {
35         kill = vent*2 - kill;
36     }
37     if (kill > 20) {
38         return x+2;
39     } else {
40         return x-1;
41     }
42 }
43
44 void shields (int z) {
45     int task = 0;
46     do {
```

```
47          task += z+task;
48          z = task % 30;
49      } while (task % z != 0);
50  }
51
52  int electrical (int y) {
53      int wire;
54      int panel = 0;
55      for (wire = 2; wire > 0; wire--) {
56          panel = (wire*2);
57      }
58      if (panel > 4) {
59          y = 57;
60          return y;
61      } else {
62          y = 85;
63          return y;
64      }
65  }
```

Solution:

```
Impostor value is: 140
Impostor value is: 139
Impostor value is: 102
Impostor value is: 102
Impostor value is: 101
Impostor value is: 85
The Impostor is: U
```