ENEE 140, Spring 2020
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	Total
Points:	16	11	11	21	41	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. 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:

```
float a = 5;
int b = 2;
char c = 'Z';
unsigned d = 6;
```

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 = a / b;	2.5
int	r2 = a / b;	**2**
int	r3 = UINT_MAX;	**-1**
int	r4 = d / a;	**1**
unsigned	r5 = a % b;	**ERROR**
char	r6 = c −−;	**'Z'**
int	r7 = (**int**) a % b;	**1**
unsigned	r8 = (b >> 1) % b;	**1**
int	r9 = (−INT_MIN) % 2	**0**

2. (11 points) This problem tests your understanding of `for` and `while` statements. Rewrite the following code using only `for` loops.

```
#include <stdio.h>
int main(){
    int a, b, c;

    b = 10;
    a = 3;

    while( a < 10 ) {
        b = b - 1;
        c = 5;

        while( b > 0 && c > 0) {
            printf("a is %d\n" , a + c);
            c = c - 1;
        }
        a += 2;
    }
}
```

Solution:

```
#include <stdio.h>
int main(){
    int a, b, c;

    b = 10;

    for( a = 3; a < 10; a = a + 2){
        b = b -1 ;
        if( b > 0){
            for(c = 5 ; c > 0 ; c = c - 1) {
                printf("a is %d\n" , a + c );
            }
        }

    }
}
```

OR

```
#include <stdio.h>
int main(){
    int a, b, c;

    b = 10;
    a = 3;
    for( ; a < 10; a = a + 2){
        b = b -1 ;
        c = 5;
            for( ; b>0 && c > 0 ; c = c - 1) {
                printf("a is %d\n" , a + c );
            }

    }
}
```

3. (11 points) This problem will test your understanding of `while loops` and arithmetic operations in C. Given the following code, write down what the output will be.

```
#include <stdio.h>

int main() {
        int num = 8795, rem, sum = 0;

        while( num > 0 ){
                rem = num%10;
                num /= 10;
                sum += rem;
        }

        printf("The output is: %d\n", sum);
        return 0;
}
```

Solution: The output is 29

4. (21 points) This question tests your understanding of character input and output. The following program should take in input from the user until they enter a newline character. For each letter the user inputs, the program prints out the reversed letter. A reversed letter is the same distance from 'Z' as the original is from 'A'. For example, `'A'` becomes `'Z'`, `'B'` becomes `'Y'`, etc.. The letter should retain its original case. If the user inputs a non-letter character, the program prints it out normally. Circle and correct the lines that contain bugs. (Hint: There are 7)

```
1  #include stdio.h
2
3  float flip_letter (char c){}
4
5  char flip_letter (char c) {
6      if (c >= 'A' || c <= 'Z') {
7          return 'Z' - c - 'A';
8      }
9      else if (c >= 'a' || c <= 'z') {
10         return 'z' - c - 'a';
11     }
12     return c;
13 }
14
15 int main () {
16     char c;
17
18     while ((c = getchar()) != '/n') {
19         printf("%c",flip_letter(c));
20     }
21     printf("\n");
22     return 0;
23 }
```

Solution:

```
Line 1: #include <stdio.h>

Line 3: float flip_letter (char c);

Line 6: if (c >= 'A' && c <= 'Z') {

Line 7: return 'Z' - (c - 'A');

Line 9: else if (c >= 'a' && c <= 'z') {

Line 10: return 'z' - (c - 'a');
```

```
Line 17: while ((c = getchar()) != '\n') {
```

5. (41 points) This is a two-part question.

   (a) The first part of the question tests your knowledge of functions and for loops.

   A perfect number is an integer N whose positive factors, including 1 and excluding itself, sum to N. For example, 28 is a perfect number because its positive factors are 1, 2, 4, 7, and 14, and 1+2+4+7+14=28. 6, 496, and 8128 are also perfect numbers for the same reason.

   The `is_perfect_number` function takes in a number `n` and uses a for loop to add up all its positive factors. If the sum is the same as `n`, it returns 1. Otherwise, it returns 0. Fill in the blanks to make `is_perfect_number` work properly.

   Bonus (+ Nobel Prize pts.): Find an odd perfect number.

```
int is_perfect_number(____int____ n) {

  int i;

  int sum = ____0____;

  for (____i=1____;____i<n____;____i++____) {

    if ((____n/i____)*i==n) {

      sum = ____sum____ + ____i____;

    }
  }

  return ____sum==n____;
}
```

Solution:

```
int is_perfect_number(int n) {
  int i;
  int sum = 0;

  for (i = 1; i < n; i++) {
    if ((n/i)*i==n) {
      sum = sum + i;
    }
  }

  return sum==n;
}
```

(b) The second part of the question tests your understanding of C syntax. The following program consists of the function prototype for `is_perfect_number`, the `main` function that uses it to tell the user whether or not their input is a perfect number, and a correct implementation of `is_perfect_number` (not shown). This program should prompt the user for an integer, store it in `a`, and print back to the user whether or not `is_perfect_number` evaluated to true. Circle and correct the lines that contain bugs. (Hint: There are 7 bugs)

(You do not have to fill in any of the blanks here. They are included to avoid giving away answers in the first part of the question).

```
1  stdio.h
2
3  float is_perfect_number(_____ n);
4
5  int main() {
6    int a;
7
8    printf("Please enter a number: ");
9    scanf("%d",a);
10
11   if is_perfect_number(a) {
12     printf("%c is a perfect number! :D\n",a);
13     else {
14       printf("%d is not a perfect number. :(\n",&a);
15     }
16   }
17
18   return 0
19 }
20
21 int is_perfect_number(_____ n) {
22   ......
23 }
```

Solution:

```
Line 1: #include <stdio.h>

Line 3: int is_perfect_number(____ n);

Line 9: scanf("%d",&a);

Line 11: if (is_perfect_number(a)) {

Line 12: printf("%d is a perfect number! :D\n",a);

Line 13-15: else block should be outside of the if block,

Line 14: printf("%d is not a perfect number. :(\n",a); // remove &

Line 18: return 0;
```