ENEE 140, Spring 2023
Final Exam — Answer Key
Do Not Make a Copy!!

Date: Saturday, May 13, 8–10 am.

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	10	6	10	12	15	7	10	14	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:

```
int a = 2;
int b = 4;
float c = 3.0;
char d = 'c';
```

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
int	r1 = b / a;	2
int	r2 = a / b;	**0**
float	r3 = a / (**int**)c;	**0.0**
unsigned	r4 = UINT_MAX + a;	**1**
float	r5 = b / c;	**1.33**
int	r6 = (a > b);	**0**
char	r7 = d − a;	**'a'**
int	r8 = a + b++;	**6**
int	r9 = (**int**)c | a;	**3**

2. (10 points) This question will test your knowledge of loops.

Rewrite the following while loop as a for loop.

```
int k=1;
while(k < 25){
    printf("%d ", k);
    k+=3;
}
```

Solution:

```
for(k=1; k<25; k+=3){
```

```
    printf("%d ", k);
}
```

3. (6 points) This question will test your understanding of structs and arrays. The following code snippet describes two structs that can be used to store information about a class of students.

```
typedef struct _student {
    int ID;
    float GPA;
    char first_name[100];
    char last_name[100];
    int year;
}student;

typedef struct _class {
    student section[20];
    char prof_name[100];
} class;

class enee140;
```

Note: In the code above the variable `enee140.section[4].GPA` holds the GPA of the fifth student.

(a) What variable name corresponds to the string holding the professor's name?

A. enee140.prof_name

B. enee140.prof_name[0]

C. enee140.section.prof_name

D. enee140.prof_name.section

(a) **A**

(b) What variable name corresponds to the string that holds the first name of the student in position n?

A. enee140.first_name[n].section[n]

B. enee140.section[n].first_name

C. enee140.section.first_name[n]

D. enee140.first_name[n].section

(b) **B**

(c) What variable name corresponds to the character holding the first letter of the first student's last name?

A. enee140.section [1]. last_name[1]

B. enee140.section [0]. last_name[0]

C. enee140.last_name[1]

D. enee140.section [1]

(c) **B**

Solution:

A. enee140.prof_name
B. enee140.section[n].first_name
B. enee140.section[0].last_name[0]

4. (10 points) This question will test your knowledge of arrays and functions.

The following function is supposed to calculate the dot product of two vectors. The function takes three arguments, two integer arrays, A and B, representing the two vectors and one integer representing the length of the vectors. It then returns a integer which is the dot product. Fill in the blanks.

Note: The dot product of two vectors is the sum of the products of each corresponding element. For example, if we had vectors of length two (a,b) and (c,d), the dot product would be (a*c)+(b*d).

```
int dot_product(int A[], int B[], int length) {
    int sum=0;

    for(int i=0; i<length; i++){
        sum += A[i]*B[i];
    }

    return sum;
}
```

Solution:

```
int dot_product(int A[], int B[], int length) {
    int sum=0;

    for(int i=0; i<length; i++){
        sum += A[i]*B[i];
    }

    return sum;
}
```

5. (12 points) This question will test your understanding of string/arrays and loops. The purpose of the following function is to copy a string from the `src[]` array to the `dst[]` array while removing any instances of a particular character passed through the remove variable. The function should also return the number of characters removed. For example the function call

   `remove_letter(dst[], "test input", 't');`

   should return 3 and `dst[]` should be `"es inpu"` after the call. Fill in the blanks so that the function works as described.

```
int remove_letter(char dst[], char const src[], char remove) {
    int i = 0;
    int j = 0;
    int count = 0;
    while(src[i] != '\0') {
        if(src[i] != ____remove____) {
            dst[____j____] = src[____i____];
            ____j____++;
        }
        else {
            ____count____++;
        }
        ____i____++;
    }
    dst[j] = '\0';
    return count;
}
```

Solution:

```
int remove_letter(char dst[], char const src[], char remove) {
    int i = 0;
    int j = 0;
    int count = 0;
    while(src[i] != '\0') {
        if(src[i] != remove) {
            dst[j] = src[i];
            j++;
        }
        else {
            count++;
        }
        i++;
    }
```

```
    dst[j] = '\0';
    return count;
}
```

6. (15 points) This question will test your knowledge of 2D arrays and file input.

   The following program reads in a square matrix from an input file. The input file in designated as the first command line argument to the program. The input file can be assumed to have an integer designating the dimension of the matrix and then the integer values of each entry, separated by spaces. The entry values go in order from left to right along rows and from top to bottom of the matrix.

   There are currently 5 bugs on 5 lines of the program. Mark down the line number and what the correct line is in the space below.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  #define MAX 100
5
6  int main(int argc, char argv[]){
7      char filename[15];
8      int num, dim;
9      int matrix[MAX][MAX];
10
11     if(argc != 2){
12         printf("Usage: a.out <filename>");
13         return 1;
14     }
15
16     strncpy(argv[1], filename, 15);
17
18     FILE* fptr = fopen(filename, "r");
19
20     if(fptr == NULL){
21         printf("Failed to open %s\n", filename);
22         return 2;
23     }
24
25     fscanf(filename, "%d", &dim);
26
27     for(int i=0; i<=dim*dim; i++){
28         fscanf("%d", &num);
29         matrix[i/dim][i%dim] = num;
30     }
31
32     fclose(fptr);
33     return 0;
34 }
```

Solution:

```
Line 6: int main(int argc, char* argv[])

Line 16: strncpy(filename, argv[1], 15);

Line 20: fscanf(fptr, "%d", &dim);

Line 22: for(int i=0; i<dim*dim; i++){

Line 23: fscanf(fptr, "%d", &num);
```

7. (7 points) This problem tests your understanding of logical operators.

   Short circuiting occurs when the result of a Boolean expression, utilizing the && or || logical operators, can be inferred from the result from the expression on the left hand side. For example, the expression: $(1 > 4)$ && $(2 < 3)$ will not evaluate the comparison on the right hand side $(2 < 3)$. The result can only be zero since the left hand side $(1 > 4)$ evaluates to 0, and 0 && x is 0 for all possible values of x.

   Hint: Think about the truth tables for both && and || operators.

   Which of the following Boolean expressions will not exhibit short circuiting when implemented in C. Assume A is 1 and B is 0.

   A. A || B

   B. B && A

   C. A && B

   D. (A < B) && B

7. C

Solution:

C: A && B

8. (10 points) This question will test your understanding of basic C syntax and program structure. Fermat's last theorem states that there are no strictly positive integers (1, 2, 3, ...) x, y, z, and n such that $x^n + y^n = z^n$, given that $n > 2$. Your mathematician friend wrote a program that checks for Fermat's last theorem on four inputs, but they have not taken ENEE140 and so their code is riddled with errors. There are five errors; repeated errors count as one error. Find all five errors, and write the line number of each error along with a correction. Assume that small enough inputs will be provided such that there are no integer ovrflows.

```
1  #include <stdio.h>
2
3  int main {
4      unsigned x, y, z, n, xs, ys, zs;
5      zs = 1;
6      ys = 1;
7      xs = 1;
8
9      printf("Enter in x y z and n:\n");
10     scanf("%u %u %u %u", x, y, z, n);
11
12     if(n < 2) {
13         printf("n is too small\n");
14         return 0;
15     }
16
17     for(int i = 1; i < n; i++) {
18         zs = zs*z;
19         ys = ys*y;
20         xs = xs*x;
21     }
22
23     if((xs + ys) = zs)
24         printf("CONGRATULATIONS!!! You found a solution!\n");
25     else if((xs + ys) > zs) {
26         printf("Not a solution x and y are too big\n");
27     }
28     else {
29         printf("Not a solution z is too big\n");
30     }
31     return 0;
32 }
```

Solution:

Line 3 Missing parentheses on `int main()`

```
Line 10 scanf is missing &s
Line 12 if should include n = 2 case
Line 17 for loop should run n times
Line 23 Should be == for equality check not =
```

9. (14 points) This question will test your understanding of functions and variable scope. Write the `EXACT` output of the following program in the answer space provided.

```
#include <stdio.h>

int func(int array[], int num) {

    for(int i = 0; i < 5; i++) {
        if(i & 1) {
            array[i] += num;
        }
        else {
            array[i] = i;
        }
        ++num;
    }

    return num;
}

int main() {
    int tracker[5] = {0, 0, 0, 0, 0};
    int num = 1;
    int x = 2;

    num = func(tracker, x);


    printf("num is: %d\n", num);
    printf("x is: %d\n", x);
    printf("Tracker is:");
    for(int i = 0; i < 5; i++) {
        printf(" %d", tracker[i]);
    }

    return 0;
}
```

Solution:

```
num is: 7
x is: 2
Tracker is: 0 3 2 5 4
```